# Hypergraph-based interconnection networks for large multicomputers

Mohamed Ould-Khaoua

A thesis submitted for the Degree of Doctor of Philosophy
to the Faculty of Science, University of Glasgow.



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

*I dedicate this thesis to my parents and brothers for their encouragment, support and understanding.*

# Acknowledgements

I would like to thank my two supervisors, Dr. L.M. Mackenzie and Dr. R.J. Sutherland, for their encouragement and modesty which they have shown throughout the course of this work, and also for their technical help, advice in the preparation and comment on the earlier versions of the thesis.

My thanks are also due to:

- Motorola Ltd. (East-Kilbride) for funding the COBRA Project.
- Professor D.C. Gilles for his assistance and suggestions.
- All the staff and fellow post-graduate students of the Department of Computing Science. In particular, the members of the COBRA group: T. Kelly, H. Stadtler, and P. Haddow for the useful discussions on many occasions.
- All my friends for their support and understanding.

Finally, my gratitude and all my respect must go to the Algerian government for their financial support.

# *Abstract*

This thesis deals with issues pertaining to multicomputer interconnection networks namely topology, technology, switching method, and routing algorithm. It argues that a new class of regular low-dimensional hypergraph networks, the distributed crossbar switch hypermesh (DCSH), represents a promising alternative high-performance interconnection network for future large multicomputers to graph networks such as meshes, tori, and binary $n$-cubes, which have been widely used in current multicomputers.

Channels in existing hypergraph and graph structures suffer from bandwidth limitations imposed by implementation technology. The first part of the thesis shows how the low-dimensional DCSH can use an innovative implementation scheme to alleviate this problem. It relies on the separation of processing and communication functions by physical layering in order to accommodate high wiring density and necessary message buffering, improving performance considerably.

Various mathematical models of the DCSH, validated through discrete-event simulation, are then introduced. Effects of different switching methods (e.g., wormhole routing, virtual cut-through, and message switching), routing algorithms (e.g., restricted and random), and different switching element designs are investigated. Further, the impact on performance of different communication patterns, such as those including locality and hot-spots, are assessed.

The remainder of the thesis compares the DCSH to other common hypergraph and graph networks assuming different implementation technologies, such as VLSI, multiple-chip technology, and the new layered implementation scheme. More realistic assumptions are introduced such as pipeline-bit transmission and non-zero delays through switching elements. The results show that the proposed structure has superior characteristics assuming equal implementation cost in both VLSI and multiple-chip technology. Furthermore, optimal performance is offered by the new layered implementation.

# Contents

# List of abbreviations

| | |
|---|---|
| AS-DCSH | asymmetric split distributed crossbar switch hypermesh |
| CS | circuit switching |
| CSH | crossbar switch hypermesh |
| CIU | cluster interface unit |
| Cube | binary *n*-cube |
| DCSH | distributed crossbar switch hypermesh |
| GH | generalised hypercube |
| MAM | multiple-accepting model |
| MIN | multistage interconnection network |
| MS | message switching |
| PCB | printed circuit board |
| PE | processing element |
| SAM | single-accepting model |
| SBH | shared bus hypermesh |
| SE | switching element |
| SS-DCSH | symmetric split distributed crossbar switch hypermesh |
| VCT | virtual-cut through |
| WR | wormhole routing |

# List of symbols

| | |
|---|---|
| $B$ | message aspect ratio (in phits) |
| $\mathrm{B}(S)$ | number of channels crossing surface $S$ |
| $b$ | size of PE chip |
| $C$ | number of channels per node per cluster |
| $c$ | size of the sphere of locality |
| $d$ | average message distance |
| $d_{sc}$ | average distance for inter-sphere messages |
| $d_{nsc}$ | average distance for intra-sphere messages |
| $D_t$ | decision time |
| $D_b$ | bus-arbitration and release times |
| $E$ | expectation |
| $H(i)$ | cluster slice $i$ |
| $h$ | Fraction of hot-spot traffic |
| $i,j$ | indices |
| $k$ | network width (or cluster size) |
| $L$ | mean message latency |
| $L_i$ | latency seen by a message at dimension $i$ |
| $L_i^B$ | latency due to bus contention at dimension $i$ |
| $L_i^{CS}$ | latency due crossbar switch contention at dimension $i$ |
| $L_i^R$ | latency due routing at dimension $i$ |
| $L_{i0}$ | latency on the first continuing channel in dimension $i$ |
| $L_{sc_i}$ | latency seen by a message at sphere dimension $i$ |
| $L_{nsc_i}$ | latency seen by a message at non-sphere dimension $i$ |
| $M$ | message length (in phits) |
| $\mathrm{M}(S)$ | maximum of channels crossing surface $S$ |
| $m$ | probability that a PE generates a message in a cycle |
| $m'$ | traffic rate on network channels |
| $m_b$ | traffic rate on a bus seen by a message at dimension $i$ |
| $m_c$ | traffic rate on continuing channels at dimension $i$ |
| $m_p$ | traffic rate that passes dimension $i$ |
| $m_s$ | traffic rate that skips dimension $i$ |
| $m_{pp}$ | traffic rate that passes dimension $i$+1 and passes dimension $i$ |
| $m_{ps}$ | traffic rate that passes dimension $i$+1 and skips dimension $i$ |
| $m_{sp}$ | traffic rate that skips dimension $i$+1 and passes dimension $i$ |

| | |
|---|---|
| $m_{ss}$ | traffic rate that skips dimension $i$+1 and skips dimension $i$ |
| $m_{pl}$ | traffic rate that passes dimension $i$+1 and comes from the left direction |
| $m_{pr}$ | traffic rate that passes dimension $i$+1 and comes from the right direction |
| $m_{sl}$ | traffic rate that skips dimension $i$+1and comes from the left direction |
| $m_{sr}$ | traffic rate that skips dimension $i$+1 and comes from the right direction |
| $m_{ppo}$ | traffic rate, originated from the remaining ($k$-2) channels, passes dimension $i$+1 and passes dimension $i$ |
| $m_{pso}$ | traffic rate, originated from the remaining ($k$-2) channels, skips dimension $i$+1 and skips dimension $i$ |
| $m_{spo}$ | traffic rate, originated from the remaining ($k$-2) channels, skips dimension $i$+1 and passes dimension $i$ |
| $m_{sso}$ | traffic rate, originated from the remaining ($k$-2) channels, skips dimension $i$+1 and skips dimension $i$ |
| $m_{ppr}$ | traffic rate, originated from the right direction, passes dimension $i$+1 and passes through the right direction at dimension $i$ |
| $m_{ppl}$ | traffic rate, originated from the left direction, passes dimension $i$+1 and passes through the right direction at dimension $i$ |
| $m_{spl}$ | traffic rate, originated from the left direction, skips dimension $i$+1 and passes through the right direction dimension $i$ |
| $m_{spr}$ | traffic rate, originated from the right direction, skips dimension $i$+1 and passes through the right direction dimension $i$ |
| $m_{ssr}$ | traffic rate, originated from the right direction, skips dimension $i$+1 and skips dimension $i$ |
| $m_{ssl}$ | traffic rate, originated from the left direction, skips dimension $i$+1 and skips dimension $i$ |
| $m_{spr}$ | traffic rate, originated from the right direction, skips dimension $i$+1 and passes through the right direction at dimension $i$ |
| $m_{spl}$ | traffic rate, originated from the left direction, skips dimension $i$+1 and passes through the right direction at dimension $i$ |
| $N$ | network size ($N=k^n$) |
| $n$ | network dimension |
| P | pin-out of a network |
| $P$ | probability |

| | |
|---|---|
| $P_t$ | probability of termination |
| $P_{t_{sc}}$ | probability of termination for an sphere message |
| $P_{t_{nsc}}$ | probability of termination for non-sphere message |
| $r$ | mean message response time |
| $S$ | surface intersecting network channels |
| $S(i,j)$ | slice $i$ contains the $j$th slice of the CIUsclice |
| $s$ | number of slices |
| $V$ | variance |
| $W$ | channel width |
| $\mathsf{W}$ | bisection width of a network |
| $w$ | mean message waiting time at an G/D/1 queue |
| $w_l$ | mean waiting time at a queue leading to the local PE |
| $w_m$ | mean waiting time at a multiplexer |
| $w_{m_{sc}}$ | mean waiting time at a multiplexer for a sphere message |
| $w_{m_{nsc}}$ | mean waiting time at a multiplexer for a non-sphere message |
| $w_o$ | mean waiting time at a queue associated with a network channel |
| $w_{o_{sc}}$ | mean waiting time for an sphere message at a output queue associated with a network channel |
| $w_{o_{nsc}}$ | mean waiting time at a output queue associated with a network channel for a non-sphere message |
| $x_i$ | node coordinate at dimension $i$ |
| $x, y, z$ | dimension name |
| $\rho$ | message traffic rate on a network channel |
| $\rho_s$ | message traffic rate from dimension $i$+1 to $i$ |
| $\rho_t$ | message traffic rate from a dimension to the local PE |
| $\rho_i^{Rest}$ | message traffic rate at channel $i$ under restricted routing |
| $\rho_i^{Rand}$ | message traffic rate at channel $i$ under random routing |
| $\rho_{sc}$ | message traffic rate on a sphere channel |
| $\rho_{nsc}$ | message traffic rate on a non-sphere channel |
| $\beta$ | degree of locality in the sphere of locality model. |
| $\delta$ | average number of continuing channels at dimension $i$ |
| $\lambda$ | average message arrival time at a queue |

# List of figures

## Figure

# 1 Introduction

Large-scale multicomputers are generally considered to be the most feasible way of achieving, in the foreseeable future, the enormous computational power required by applications in science, engineering, and a number of other fields [1-8]. There are many software and hardware issues which stand in the way of the successful construction of such machines and a considerable research effort has been mounted in recent years to resolve these. A multicomputer is a parallel machine consisting of several *processing elements* (PEs) which co-ordinate their activities to solve a common problem, communicating by means of an *interconnection network*. This network is a critical determining factor in the overall performance of the system and is constructed from *switching elements* (SEs) and *channels*: the SEs are responsible for moving information from one PE to another via the channels. Each SE is typically associated with one PE: The PE-SE assembly is generally referred to as a *node*.

The communication pattern in the network depends strongly on the application being executed and has a great impact on network performance. If multicomputers are to gain the wide acceptance of Von Neumann machines they must be *general purpose*, that is, able to execute a wide variety of applications with clear cost-effective performance gains. It is true that the full realisation of this goal is contingent on developments in programming languages, compiler design and, perhaps most importantly, in the creation of a general abstract model of parallel machines enabling the same decoupling of high and low-levels issues as in sequential systems [9, 10]. Ultimately, however, the concept

of a general purpose multicomputer relies on the ability of its network to handle a broad spectrum of communication patterns corresponding to a wide range of potential application software. In addition to the communication pattern, the other main features which affect the network performance can be categorised into four main areas: *topology*, *technology*, *switching method* and *routing algorithm.*

## 1.1 Topology

The network topology defines the way nodes are connected. Graph theory is a useful tool in the description of network topologies, where each node corresponds to a vertex and each channel to a directed edge [11]. It is conventional to refer to a network and its corresponding graph as if they were identical.

An important distinction may be drawn between *graphs* or *hypergraphs* [12]. In a graph, each edge connects only two vertices, whereas in a hypergraph there is no such restriction. Two vertices $x$ and $y$ that are directly connected by an edge are said to be *adjacent.* Two vertices connected indirectly by a sequence of intervening edges are then said to be *non-adjacent.*

A topology defined in this way may have certain important characteristics. The *degree* of a node is the number of its adjacent nodes and the *degree of a topology* is the maximum degree of any node. The *diameter* of a topology is the maximum value of the shortest distance over all pairs of nodes. Finally, the network is said to be *regular* if all nodes have the same degree.

Ideally, a network topology should have a small number of edges, a small degree, a low diameter, and regular structure. Needless to say, a large variety of topologies have been suggested in the hope of approaching these goals [13-20].

## 1.2 Technology

The *bandwidth* of the network channels is a critical factor in network performance. The bandwidth depends on the channel *width*, i.e. the number of parallel wires each of which transfers a bit at a time, and on the channel *cycle time*, the time required to transmit a bit. With higher channel bandwidth, messages are transmitted more quickly, reducing the possibility of channels getting saturated. Implementation technologies impose constraints on both these factors [21-27]. While the effects on channel cycle time will be examined later, the following discussion focuses on the channel width constraint.

Where the channel width constraint has been ignored, network graphs have typically been compared assuming very wide communication channels which allow messages to be transferred from one node to another in a single cycle [28-33]. However, this is not usually realistic. A topology with a rich interconnection pattern is likely to have channels with lower bandwidth than one with a simpler interconnection pattern because the former has more channels and therefore will have fewer wires per channel. Furthermore, due to the limited channel width, a message has to be broken into several phits (channel words) each of which is transferred in one cycle; if the channel width is $W$ bits, a message of length $M$ bits must be broken up into $B=M/W$ phits each of which contains $W$ bits [22, 23, 34]; $B$ will be referred to as the *message aspect ratio.*

The essence of the problem is as follows. When any topology is implemented, it must be mapped into three-dimensional space. The bandwidth across any surface, $S$, intersecting the space occupied by such a topology is limited by the implementation technology to some maximum value, say, $B(S)$ (Figure 1.1). A channel can only carry a limited amount of data, generally due to limitations of the transmitter and receiver, but ultimately due to the finite bandwidth of the channel itself. Furthermore, the number of such channels which can cross $S$ is also limited because the channels themselves have non-zero physical cross-sectional area. For instance, not only do wires have finite thickness, but they cannot even be packed together as densely as their physical dimensions allow because of problems such as crosstalk interference and low line impedance [35].

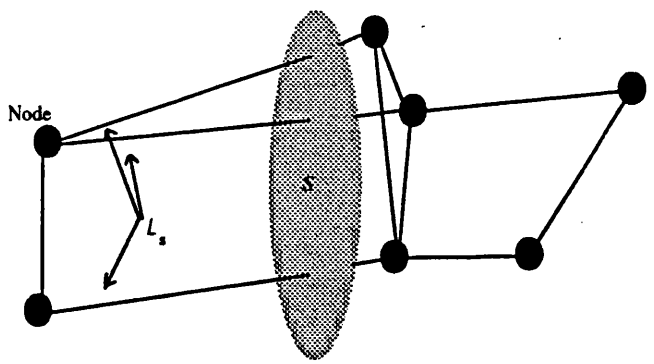



**Figure 1.1: Surface, $S$, intersecting a network imposes bandwidth constraints on $L_s$.**

Let

$$L_S = \{l_1, l_2, l_3, \ldots l_n\}$$

be the set of channels which cross $S$. $L_S$ has associated with it a maximum possible bandwidth $\mathsf{M}(L_s)$ which is certainly no greater than the sum of the maximum receiving bandwidths of the nodes attached to each of the $l_i$. If

$$\mathsf{M}(L_s) > \mathsf{B}(S)$$

then the surface $S$ imposes a bandwidth constraint on the implementation.

It is possible that some implementation of a topology might experience no bandwidth constraints, in which case performance would be limited by the speed of its nodes; however, this situation only arises when a communication technology is available which has very high capacity relative to the PEs. In high performance electronic systems, whether or not they are implemented on a single chip, bandwidth constraints are generally a critical limiting factor, but the way in which these constraints manifest themselves, and consequently their impact on different topologies, is highly dependent on the interconnection technology employed. Here the two practical forms of bandwidth constraint which have been examined most extensively in the literature are reviewed.

### 1.2.1 Wiring density

In a two-dimensional interconnection system, such as on-chip VLSI or printed circuit board (PCB), a topology must be mapped into a 2-D rather than a 3-D space, and the intersecting surfaces discussed above degenerate to lines which cut the area occupied by the flattened topology. Because wire dimensions must exceed the minimum cross-section and inter-wire spacing limits dictated by the given technology, any such intersecting line can cut at most a maximum number of wires per unit length, a characteristic of that technology called its *wiring density*.

As a convenient approximation, Dally has used the *bisection width* of the network, i.e. the number of wires cut when the interconnection network is divided in half [36], as a measure of wiring density in his evaluation of various network topologies belonging to the family of $k$-ary $n$-cubes [22, 37, 38]. His analysis has shown that given constant wiring density, a low-dimensional high-diameter network such as a 2-dimensional torus has superior performance characteristics to a higher-dimensional low-diameter topology

such as a binary *n*-cube. Under constant wiring density, low-dimensional networks have higher channel widths than their high-dimensional counterparts because the former have simpler interconnection patterns and therefore can support more wires per channel.

The results of the studies carried out by Dally have had a considerable influence on the design of experimental and commercial large-scale multicomputers. The high dimensional binary *n*-cube which was used in the iPSC [39], iPSC/2 [40] NCUBE *N*/10 [41], and Cosmic Cube [42] has fallen out of favour to be replaced by the low-dimensional torus or mesh in more recently designed machines such the J-Machine [43], Symult 2101 [44], iWarp [45], the Stanford Dash Multiprocessor [46], Horizon [47], and Delta-Touchstone [48].

### 1.2.2 Pin-out

It can be argued, however, that this move towards low-dimensional structures is somewhat premature. The wiring density argument is certainly applicable where an entire network is implemented on a single die (or PCB), but is not applicable in the more realistic situation where a network has to be partitioned over many chips (or PCBs). In such situations, the most critical bandwidth constraints are imposed by those surfaces which form the perimeter of a node and across which any data entering or leaving that node must travel. Where a node is implemented as a single die with conventional electrical inter-chip signalling, this corresponds to the number of pins available. Abraham *et al* [23, 34, 49] have studied *k*-ary *n*-cubes under a constant pin-out rather than constant wiring density constraint and concluded that high-dimensional low-diameter networks such as the binary *n*-cube perform better than low-dimensional high-diameter counterparts.

### 1.2.3 Other technological considerations

Dally and Abaraham *et al* have thus arrived at opposite conclusions about the relative performance of binary *n*-cube and torus structures simply by choosing different technological assumptions. Yet even when the channel width constraints applicable to a particular implementation technology have been identified, assumptions about a number of issues can greatly influence the outcome of any comparative study. For instance, the channel cycle time, the SE *decision time*, and the *relay bandwidth* can have a significant effect on the relative performance of high-diameter networks [50-53].

- It is often stated that longest wire length is a fundamental limiting factor in determining maximum network channel cycle time [1, 23, 37, 51]. However this assumes the use of handshaking protocols in data exchange between a source and destination. In [23, 37, 52], the conclusions reached about the optimal *k*-ary *n*-cube topology for a given network size are based on this assumption. If wires with transmission-line characteristics are employed, the wire itself has a storage capacity and can simply be treated as a sequence of stages in the pipelined transfer of bits; there is no need to wait for one phit to arrive before transmitting the next one [51, 53]. Scott *et al*, assuming pipelined transmission, found a significant effect on their model [53]. (The analyses, presented in the following chapters, assume pipeline-bit transmission which lower the effect of long wire delays).

- The decision time delay (i.e., how many cycles required to propagate a message from input to output) is affected by the way in which the SE operates, the basic gate delays of the technology of fabrication and of course the internal switch complexity. Both Dally and Abraham *et al* omitted this factor in their analyses. However, recent studies have shown that when this delay factor is taken into account the optimal network dimension is affected [51, 53]. It is worth mentioning that even the best decision times achievable at present are comparable with the time taken to drive several metres of wire; typical decision times are of the order of tens to hundreds of nanoseconds while typical wire delays are of the order of only a few nanoseconds [54, 55].

- The primary function of the SE is to switch a message from an input channel to an output channel, bringing it closer to its destination. The relay bandwidth of an internal path, relaying an input channel to an output channel can hinder the exploitation of any possible high wiring densities and pin-out counts [50, 56, 57]. This inefficiency arises when the SE has several switching paths requiring high bandwidth to match that of wide network channels. For instance, as will be discussed later, several topologies such as meshes and tori can in theory afford wide network channels when implemented in VLSI or multiple-chip technology. However, in practice these channels cannot be made very wide due to the difficulty in providing the same relay bandwidth inside the SEs as outside them.

## 1.3 Switching methods

A message is usually composed of a *header*, one or two phits which contain routing information such as destination address, and a *body* which comprises data to be transferred. A switching method determines the way by which message phits visit intermediate SEs. The commonest methods are *message switching* [58], *virtual-cut through* [59], *circuit switching* [58], and *wormhole routing* [60] (the term "routing" here should not be confused with its usage in the next section).

With message switching (MS), a message has to be buffered entirely at each intermediate SE during its journey. This method therefore suffers from a high overhead associated with message buffering. This overhead increases with the message length and distance. Early commercial multicomputers used MS [39-42].

Virtual-cut through (VCT) is a more efficient scheme which avoids buffering overhead. When a header reaches an intermediate node, it is buffered in order to decide which output channel to forward the message through. After the routing decision is made, if the required output channel is not busy, then the data phits need not be buffered but are transmitted to the next node as they arrive in a pipelined fashion. VCT has clear theoretical advantages over MS but requires a more complex and costly SE and has therefore not been much used in practice.

In circuit switching (CS), only when a complete path has been established from source to destination does message transfer start. After transfer is completed, the path is released. Examples of multicomputers that use CS are the Mark-IIIe hypercube [61] and the iPSC2 [62].

Each of these methods has advantages given certain traffic conditions [63]. MS works well for short messages and when traffic is sporadic. CS provides good performance for long messages and under light traffic. VCT combines the advantages of both MS and CS but at a significant additional implementation cost. Under light traffic, VCT behaves like CS because messages can cut-through intermediate SEs. Under high traffic, on the other hand, VCT behaves like MS as the opportunities for cutting-through vanish, and therefore messages are buffered at almost every intermediate SE.

In MS, CS, and VCT, although the message is divided into several phits, the transfer and

flow control between adjacent nodes is performed at a message level (e.g., several phits). In wormhole routing (WR), a message is broken into units called *flits*, each of which typically consists of one or two phits [37, 38]. These flits form the smallest unit of information on which transfer and flow control is performed. When the header flit (which contains the entire message header) reaches an intermediate SE, it is buffered. After a routing decision is made, if the required output channel is not busy, data phits are transmitted to the next SE as they arrive, establishing a pipeline through the network. When the required output channel is busy, the entire message is not buffered; instead the flits occupy network channels that they have already acquired. Since only the header contains routing information, these flits must remain in contiguous channels and cannot be interleaved with flits of other messages. When the header flit is blocked, all the body flits of the message stop advancing and block the progress of any other message which require the channels they occupy. When the last flit crosses an intermediate SE, the output channel is released and eventually used by other messages.

WR can be considered as an efficient variation of CS. Instead of waiting for the path to be established from source to destination as it is in CS, data transmission starts as soon as the header flit advances through the network. It is possible for the header flit to arrive at the destination before the last flit has left the source. Also, unlike CS, the circuit is released progressively as the last flit crosses a given SE.

WR has several advantages. The amount of storage required at each SE is very small in that buffers of only a few phits capacity are necessary. If the network is intended for applications that generate light traffic, WR behaves like VCT. However, WR is more cost-effective because it makes the SE simpler to implement and runs faster due to the reduced complexity of the circuits [64-67]. Unfortunately, this minimal storage approach results in lower performance than VCT when traffic is heavy.

WR is particularly advantageous for high-diameter networks operating under light to moderate traffic because latency is insensitive to the message distance. It is widely used in currently available multicomputers such as the Symult series [44], J-machine [43, 68], Delta-Touchstone [48], iWarp [45], and T9000/C104 [69], which are based on meshes and tori and support coarse and medium-grain applications which generate light and moderate traffic.

## 1.4 Routing algorithms

In a network, a routing algorithm defines a sequence of SEs and communication channels that a message has to visit between source and destination. There are several routing strategies that can be employed notably *restricted routing*, *random routing*, and *adaptive routing* [70, 34, 71].

In restricted routing, the route a message has to take is solely determined by the address of the source and destination nodes, and no network state information is taken into account when making routing decisions at SEs. A message is not allowed to change its route even though the network topology may provide more than one such route. In random routing, when there are multiple routes, one such route is selected randomly. In adaptive routing, a message can change its route during its journey according to the current network state. For instance, a message may change its route to avoid heavily congested regions [71-74]. Most commercial multicomputers use restricted routing because it is simple to implement and reduces delays through SEs [37, 65, 66, 75, 76].

Because a network has limited resources, it is important to ensure that a routing algorithm is *deadlock-free* so that a message is not delayed indefinitely in the network and never delivered to its destination [58]. Given that a network can contain cycles in its graph, a set of routes will be deadlocked if each route holds resources needed by another route while waiting for resources owned by yet another, causing a dependence cycle. Deadlock avoidance is ensured by techniques such as releasing every resource acquired or by avoiding dependency cycles when acquiring resources.

Switching methods that operate at a message level such as MS, VCT, and CS have used mainly the *structured-buffer-pool* algorithm to avoid deadlock [58, 77, 78, 79,80]. In this algorithm, at each node message buffers are partitioned into classes, and the assignment of buffers to messages is restricted to define a partial order on buffer classes. Since the minimum number of buffer classes is equal to the diameter of the network, the number of buffers necessary to maintain this structure tends to grow with the size of the network, an undesirable feature for large-size multicomputers.

In WR, the most common deadlock avoidance algorithm involves the use of *virtual channels* [22, 81, 82]. Each virtual channel, which has its own dedicated buffers, is time-multiplexed with other virtual channels onto a physical channel. Deadlock is avoided by selecting a unique sequence of virtual channels when routing messages from source to

destination. The restricted routing approach, known as the *e*-cube, proposed by Barkshow [83], is widely used in most commercial multicomputers and is a special case of the virtual channel algorithm where there is a one-to-one mapping between virtual channels and physical channels.

## 1.5 Motivations

The short history of parallel systems can be divided into three generations. The first corresponded to the use of systems based on the shared memory model, known as *multiprocessors* [84-86]. However, due to the problems with scaling multiprocessors to even moderate sizes [4, 52], multicomputer systems supporting the message-passing programming model became more popular. The first characteristic of the second generation systems is their reliance on locality, which is fortunately an important feature in many applications that fit in the message-passing model. As these early multicomputers used MS, low diameter networks like binary *n*-cubes were needed to reduce the buffering overhead of this switching method.

The most recent generation of parallel machines, also multicomputers, is based on meshes and tori which have replaced binary *n*-cubes, influenced to some extent by the work of Dally. As has been noted, his results are mainly applicable when entire systems can be accommodated on a single chip. However, this level of integration appears some way off in the future, so the question remains as to why these structures have dominated recent thinking. There are several factors: firstly, these networks are relatively easier to build than binary *n*-cubes and exhibit apparently[1] superior modularity; secondly, as there are many scientific and engineering applications that exhibit locality, meshes and tori seem natural hosts for such applications; thirdly, the adoption of wormhole routing and medium-grain parallelism has reduced the effect of long diameters.

Demands are likely to grow in the future for systems which support efficiently both local and non-local communications as the parallel processing field tackles new research areas [87, 2, 4]. Furthermore, as future multicomputers are targeted at fine-grain applications, which can harness the full benefit of parallelism, more emphasis will be put on networks which can cope with high traffic conditions [7, 8, 22, 37]. This thesis argues that a new class of regular low-dimensional hypergraph networks, referred to as *distributed crossbar switch hypermeshes* (DCSH for short), are compatible with these requirements

[1] The system cannot be scaled without increasing the channels bandwidth to maintain good performance.

and are therefore ideal candidates for high-performance interconnection networks in future multicomputers.

In graph network implementations, a node includes a PE and SE [22, 28, 34, 37, 51, 52]. However, node-based partitioning of a given topology does not necessarily give it maximal use of the available integration levels and pin-out counts. It will be shown that due to their inherent topological properties, low-dimensional DCSHs in particular may benefit from separating switching and processing functions rather than simply attempting to place one or more nodes on a single chip. This separation effectively can alleviate both wiring density and pin-out constraints.

A great deal of analytical attention has been paid to graph-based networks such as meshes and binary $n$-cubes, while hypergraphs have escaped evaluation. For instance, hardly any work has been reported in the literature which investigates the impact of switching methods and routing algorithms on the performance of hypergraphs. Unlike most previous research related to hypergraphs [88-93], this work takes important technology-related issues into account. In particular, implementation cost is considered for both on-chip VLSI and multiple chip technology. Further, the delay due to the decision time is also taken into account. It will be shown that this factor can affect the outcome considerably.

Although this thesis focuses on low-dimensional DCSHs, since they can take advantage of the new implementation scheme, their higher-dimensional counterparts will also be considered in order to fully understand the performance behaviour and assess the benefits of multi-dimensional regular hypergraphs.

## 1.6 Outline of the thesis

The investigation of the DCSH is organised into several chapters. Chapter 2 briefly reviews existing graph as well as hypergraph networks. It discusses the limitations of the previously proposed implementation schemes for hypergraph networks. It then presents the implementation proposal for a low-dimensional DCSH which allows relaxation of the bandwidth constraints yet enables it to retain a totally regular and compact form.

Chapter 3 investigates the performance of the DCSH under different switching methods and routing algorithms. Furthermore, it evaluates different switching element design alternatives and some variations of the DCSH. It also examines non-uniform traffic in the

DCSH and evaluates the performance gain that may be obtained by exploiting locality.

In Chapter 4, the other implementation schemes for regular hypergraph networks are compared with the DCSH scheme taking into account the different bandwidth constraints when implemented using VLSI and multiple-chip technology.

Among those networks that have been studied extensively in the literature are meshes and multistage networks. Chapter 5 and 6 compare these networks with the DCSH. Further, since low-dimensional DCSHs are well-suited for layered implementation, the second part of Chapter 6 assesses whether they have more performance merits than their higher-dimensional counterparts when they are implemented in VLSI and multiple-chip technology. Such comparisons take into account important technological considerations that have been ignored in the literature, such as the use of pipeline-bit transmission (which lowers the effect of long wire delays) and the effect of decision time through switching elements (which is expected to dominate wire delay for the foreseeable future).

Finally, chapter 7 presents a summary of the results obtained during the investigation along with a discussion of possible avenues of further research.

# 2 Distributed crossbar switch hypermeshes

## 2.1 Introduction

Naively, one might expect an ideal multicomputer network to be completely-connected, with each node having a dedicated connection to every other node. In such a structure, the communication time between any two nodes is the same. Due to the high pin-out per node and the VLSI area required to implement the communication-related hardware, such networks are precluded from use in practice. The goal is therefore to design a network that is feasible, scales to large sizes, and yet provides performance which is comparable in practice to that of a completely-connected network.

Several networks have been proposed in the literature [3, 6, 84, 86, 93]. Network topologies can be broadly categorised as graphs or hypergraphs depending on the interconnect structure used to realise the direct connection among the nodes. In the former an edge connects only two vertices, while in the latter an edge can connect more than two vertices. However, as will be discussed later, such a categorisation of topologies is not exhaustive in that it excludes multistage-interconnection networks [84, 85, 86, 94].

## 2.2 Graph networks

In this category, the interconnect structure is a point-to-point channel which can directly connect only two nodes [12]. Communication time depends on the relative position of the communicating nodes since communication between adjacent nodes is faster than that between non-adjacent nodes.

An example of such structures is the *ring* where a node is connected only to its left and right neighbours. This is easy to implement, but its diameter grows linearly with the total number of nodes, $N$, degrading performance as $N$ becomes large.

To reduce the diameter, networks are often organised as multi-dimensional *Cartesian products of simple basic graphs like the ring*. The $n$-dimensional product of the $k$ element ring is called $k$-ary $n$-cube, a family which includes the torus and binary $n$-cube. A variation are the meshes, which are products of simple one-dimensional "lines" of nodes.

In the early multicomputers, the binary $n$-cube seemed an optimal structure because its smaller diameter ($O(LogN)$) compared to the mesh and torus reduced the buffering overhead associated with message switching. However, for reasons discussed in Section 1.2.3, the 2-D and 3-D mesh and torus have recently tended to predominate [45-48]. While these latter structures are easy to build and exhibit good performance when wormhole routing is employed, it is important to appreciate that they also have some significant drawbacks.

### 2.2.1 Disadvantages of 2-D meshes and tori

Meshes and tori are inherently suited to regular nearest neighbour communications but have difficulties with traffic patterns involving a high proportion of non-local traffic or requiring broadcast operations such as those found in matrix computations [105, 106]. To overcome such difficulties, some authors have suggested that these structures be augmented with buses [95-97]. Further, due to the high diameter, a message on average crosses a large number of SEs, and these currently have significant switching delays (i.e., decision times). This results in long network transit time. To reduce the effect of the switching delays, Dally has proposed a variation of $k$-ary $n$-cubes called *express cubes*, in which a message can bypass several SEs by means of "express" channels [55].

Chittor has shown that although meshes and tori can have high bandwidth channels, their simple interconnect patterns demand careful mapping of process graphs to reduce the effect of high channel contention [98-100]. Further, even when optimal mapping is used, certain communication patterns, such as FFTs, cannot take advantage of the high bandwidth channels due the inherent mismatch between the communication pattern and the mesh and torus structure.

In graph networks the node, comprising a linked PE and SE, seems a natural physical building block, but this approach may not necessarily lead to the optimal physical partition of the network in order to take best advantage of the wiring density or chip pin-out count offered by the implementation technology. Athas, for example, has suggested separating processing and switching functions to increase channel bandwidth in a mesh-like structure [56]. However, his particular approach could not overcome the difficulty of providing equal relay bandwidth inside the SEs so as to harness any benefits of wide channels.

## 2.3 Hypergraphs and hypermeshes

Hypergraphs offer a more useful generalisation than graphs in that they can represent networks with diameters which grow more slowly with the number of nodes [90, 91, 92]. Several interconnect structures have been used to connect directly more than two nodes, most notably buses and crossbar switches. Complete connections or even multi-stage networks can also emulate hypergraph edges since they can provide universal connection to a group of more than two nodes.

"One-dimensional" networks based solely on a shared bus or crossbar switch can only be scaled to a limited degree, so as with graphs, it is natural to construct multi-dimensional regular hypergraphs. These are analogous to $k$-ary $n$-cubes, and are called *hypermeshes* [87, 101]. In a $k^n$ node hypermesh, each of the $n$ dimensions contains $k$ nodes (referred to as a *cluster*) which are directly connected. Figure 2.1 illustrates a 4×4 2-D hypermesh, where cluster nodes are connected by a shared bus. Hypermeshes have several desirable properties: they efficiently support many communication patterns and can even emulate some multi-stage interconnection networks [87, 89, 102]

Process graphs which map naturally onto binary $n$-cubes, binary trees and 2-D meshes can also be mapped efficiently on hypermeshes. Adjacent nodes in the process graph are

embedded into adjacent nodes in the hypermesh. In contrast, mapping the binary $n$-cube and binary-tree graphs onto the 2-D mesh and torus do not preserve such adjacency. Moreover, although 2-D meshes are very well suited for nearest neighbour communication, when 3-D mesh process graphs are mapped on such topologies, adjacent process nodes are not in general on adjacent mesh nodes [103]. Such situations do not arise when 3-D meshes are mapped onto 2-D hypermeshes.

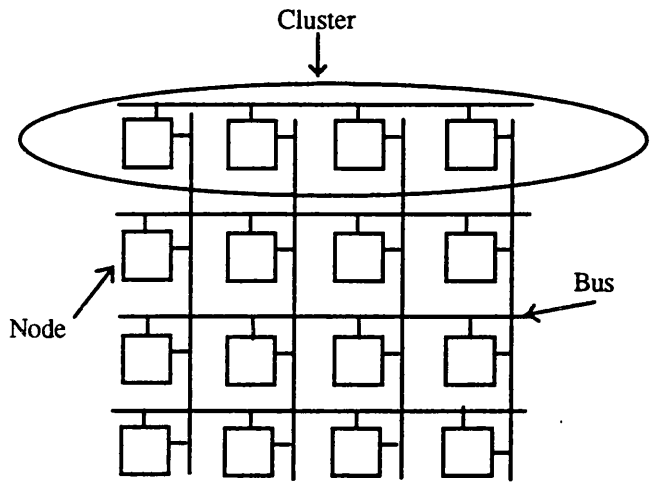



**Figure 2.1 : A 2-dimensional hypermesh.**

Hypermeshes can efficiently perform important operations such as broadcast [105, 106]. They can realise permutations of Omega and inverse Omega SIMD permutations in one pass and minimum distance [87, 89, 102, 103]. Further, they can realise all the SIMD permutations, such as those found in FFTs, which can be performed efficiently on binary $n$-cubes.

### 2.3.1 Problems of hypermesh implementations

Hypermeshes, using shared buses, were proposed first by Wittie who called them "spanning-bus hypercubes" [107]. While this implementation scheme is theoretically workable, it suffers in practice from the problems associated with shared-buses; in addition to the practical limit on the bandwidth of a shared bus, there is a significant overhead associated with frequent changes of mastership in high speed applications [108].

Crossbar switches, on the other hand, are an attractive implementation scheme because they are available commercially and so offer convenient cheaper implementation for hypermesh clusters of moderate sizes [90, 92, 93] (this implementation scheme will be referred to as the crossbar switch hypermesh). Hypermeshes using complete connection in each cluster have been proposed by Agarwal *et al* who called them "generalised hypercubes" [109, 110]. The high node degree and large number of channels, however, makes them costly to implement.

More recently, several authors have suggested the use of optical technology, such as the use of two-frequency-multiplexed star-couplers, offering high communication bandwidth [88, 89, 91, 111]. However, as integrated single-chip PEs become feasible, it will be necessary to develop opto-electronic transmitters and receivers capable of operating at 10-100 GHz rates which will not cause a disproportionate increase in the site area occupied by the PE. Commercial technologies with these parameters appear to be some way off [112].

## 2.4 Distributed crossbar switch hypermeshes (DCSHs)

In the DCSH, cluster nodes are connected by means of *distributed crossbar switches* where the multiplexing/de-multiplexing functions of the conventional crossbar switch are performed in the switching elements of the cluster nodes [50, 57, 103]. Figure 2.2 shows the basic structure of a DCSH cluster. Within a cluster, at each of the $k$ destinations, there is a ($k$-1)-to-1 multiplexer with buffered inputs. Every node possesses a uniquely-owned channel which connects it to one of the multiplexer inputs of each other node in the cluster. When $k$=2, the DCSH collapses to the binary $n$-cube [42, 113].

Let us define formally the connections between the $k^n$ nodes in the DCSH. Let the nodes be named $N_{x_1x_2...x_n}$ for $(0 \le x_1,x_2,....,x_n < n)$. If the connection between any two $N_{x_1x_2...x_n}$ and $N_{x_1'x_2'...x_n'}$ is denoted by

$$N_{x_1x_2...x_n} \sim\sim N_{x_1'x_2'...x_n'}$$

then the connections between the nodes in cluster $i$ $(0 \le i < n)$ is denoted by

$$N_{x_1...x_{i-1}x_ix_{i+1}...x_n} \sim\sim N_{x_1...x_{i-1}x_i'x_{i+1}...x_n}, \text{ for } (x_i' \ne x_i),$$

$$(0 \le x_i' < n),\ (0 \le x_1,x_2,....,x_n < n)$$

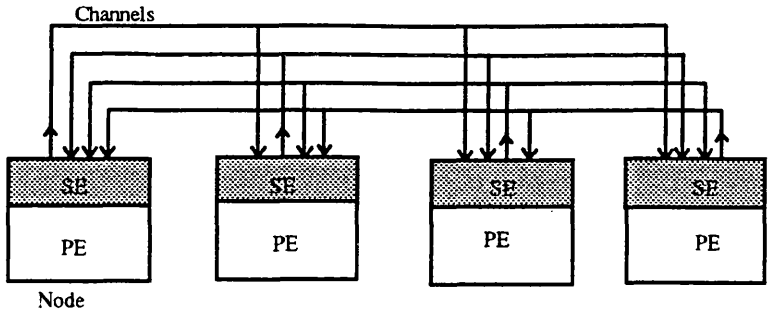



**Figure 2.2: A DCSH cluster.**

The DCSH has significant advantages over the other hypermesh implementations.

a) Distributed crossbar switches allow the support of important communication operations such as one-to-all and all-to-all broadcast within a cluster [114]. Such operations cannot be performed on a crossbar switch hypermesh.

b) The DCSH retains some of the desirable topological properties of the generalised hypercube, such as a low diameter, but requires far fewer channels and lower node pin-out, and therefore is cheaper to implement. Yet the DCSH is more efficient than the generalised hypercube in its use of bandwidth, and consequently can attain higher performance for a given cost (as will be shown in later chapters).

c) Distributed crossbar switches, as Section 2.4.1 will show, enable the DCSH to mitigate the normal bandwidth limitations imposed by restricted channel width. While all existing hypermeshes and graph networks suffer from bandwidth constraints, the DCSH can take advantage of a separation of switching and processing functions to greatly relax such constraints (Figure 2.3).

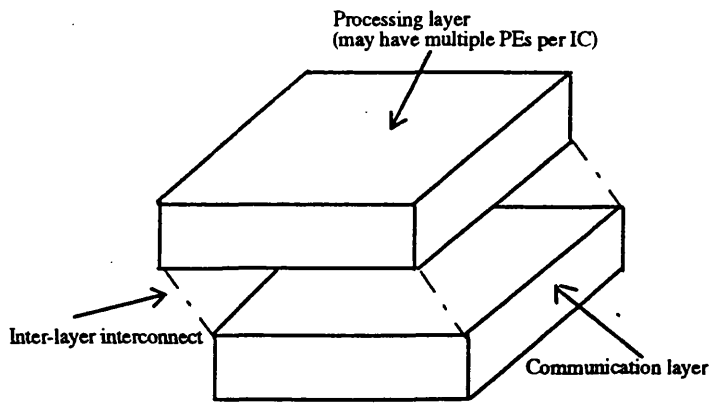



**Figure 2.3: Separation of processing and communication functions by physical layering.**

In the DCSH, a node is separated into two parts: a processing element (PE) which contains the general purpose processing and storage capability of the node and a switching element (SE) which performs all the network-dependent functions of the communication system (Figure 2.4). The two are interconnected by a PE-SE interface. The SE contains one *cluster interface unit* (CIU) for each dimension to which the node interfaces.

In an $n$-dimensional DCSH. a message will travel up to $n$ hops encountering intermediate SEs only when it changes dimension and when it is finally delivered to the destination PE. Thus in a 2-D system a message encounters at most one intermediate SE, whereas in a 3-D system it encounters at most two. At an intermediate SE, the message is transferred directly from the CIU on which it arrives to the CIU on which it leaves: there is no involvement of the PE.

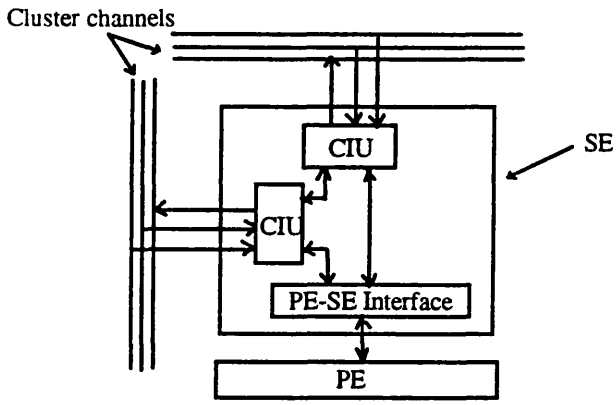



**Figure 2.4: A DCSH node structure.**

The $n$ dimensions of the DCSH system are numbered from 1 to $n$, imposing a total order on the list of clusters of which any particular node is a member. For a given node, the first, second, and third members of this ordered list will be called respectively its $x$-cluster, $y$-cluster, and $z$-cluster. The conventional addressing system used in a $k$-ary $n$-cube can also be used for a $k^n$ node DCSH, where a node address is an $n\times1$ vector $[x_i]$ with element $x_i$ being the position of the node in its cluster $i$. This vector can be uniquely characterised as a single numerical value by forming

$$\sum_{i=1}^{n} x_i k^i \qquad (2.1)$$

### 2.4.1 The DCSH implementation

The major implementation difficulty with the DCSH is the presence of non-local connections which can create problems in 2-D wiring technologies. This section presents an implementation scheme that relaxes the restrictions imposed by the wiring density constraint while minimising the constraints imposed by limited pin-out. Although the scheme is only suitable for 2-D and 3-D DCSH systems (basically because these map naturally into the physical dimensions), it can accommodate moderately large cluster sizes of 32, 64 nodes so that systems of any reasonable number of nodes can be constructed. However, the crossbar switch hypermesh implementation which is the commonest implementation scheme and have widely been reported in the literature cannot use clusters beyond 16 nodes due to the limited bandwidth of the commercial crossbar switches [90, 92, 93].

The scheme is illustrated for the 2-D DCSH but it can be extended for the 3-D case. The idea is to use the regularity of a grid layout of nodes (identical to that of a 2-D mesh) to minimise the physical size of the system, while separating the processing and network functions into layers as mentioned above. One or more PEs can be integrated on a single chip. These chips are then laid out in a two-dimensional grid of side $k$, called the *processing layer* (Figure 2.5). Each row and each column will be a cluster. Below the processing layer is a second layer with all the $x$-dimension connections and CIUs; above is a third layer with the $y$-dimension connections and CIUs. These will be referred to as the $x$ and *y-communication layers* respectively.

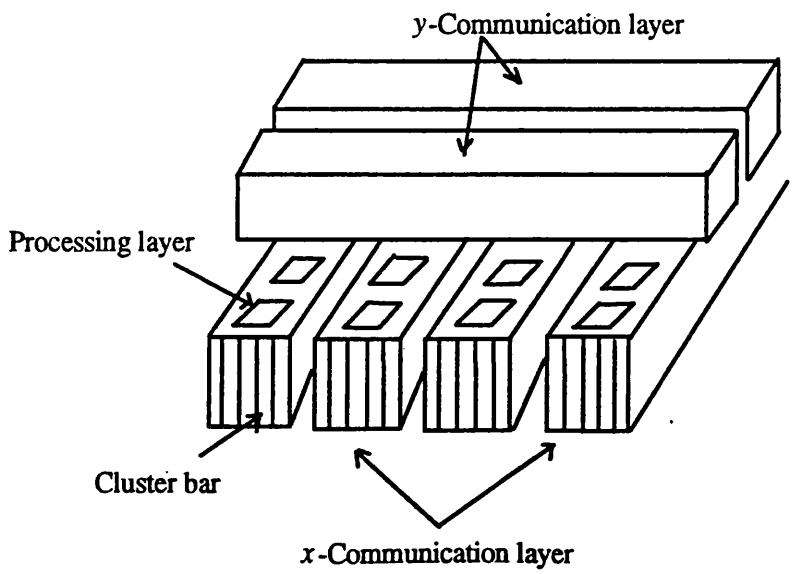



**Figure 2.5: A Layered DCSH implementation**

Consider a typical $k$ node cluster with channels, each having $W$ wires across. The CIUs interfacing those channels are bit sliced, say $s$ ways, where $W/s$ should be large enough to accommodate a row or column address and a strobe (say 9 parallel bits); this means that an individual CIU slice can perform address detection on its own, minimising the necessity for communication between the slices. Let the $j^{th}$ slice from node $i$ be called $S(i,j)$. $S(i,j)$ then contains a $W/s$-way ($k$-1)-to-1 multiplexer with input buffers, an arbiter and some control logic, including address detection. For each $j$, all $j^{th}$ CIU slices $S(i,j)$ with $i$=1..$k$ are mounted on a hybrid substrate to form a *cluster slice*, $H(j)$. All $s$ cluster slices $H(1)..H(s)$ are then oriented as shown in Figure 2.6 and aligned together as in Figure 2.7. The consequent structure, called a *cluster bar*, contains all wiring and CIUs for a single cluster and forms the basic unit of construction of the communication layers. Figure 2.5 shows that when the system is oriented in the plane of the page, so that the $x$-cluster bars run left-right and $y$-cluster bars will run top-bottom.

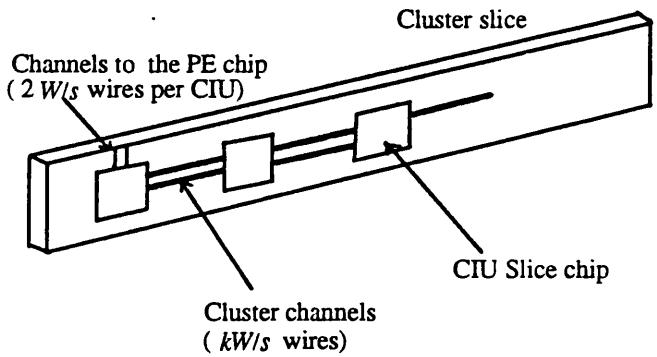



**Figure 2.6: A DCSH cluster slice**
***s*=number of slices**
***W*=single channel width**
***k*=cluster size.**

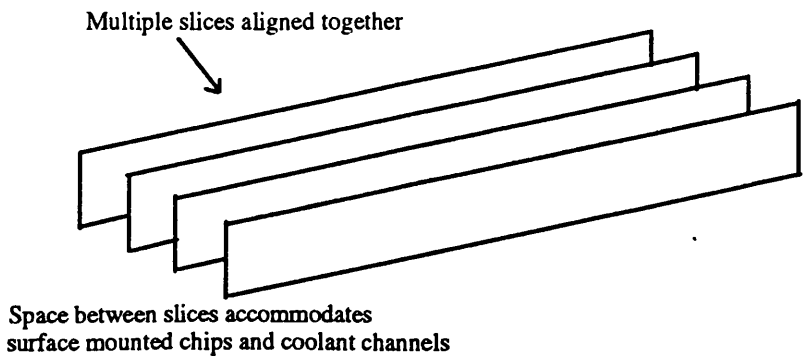



**Figure 2.7: Construction of a cluster bar.**

On the surface of a cluster slice, the main cluster channels run as traces along the length, with intervening repeaters inserted, if necessary, at intervals calculated to optimise signal travel time [115]. The interspacing of such repeaters depends on the dispersive properties of the conductive traces themselves: at current hybrid wiring densities they may not be necessary at all [116].

While a cluster slice has $kW/s$ wires along its length, each CIU requires only $2W/s$ wires ($W/s$ in each direction) to communicate with the processing layer, and, for efficiency another $2W/s$ lines connecting it to a complementary CIU in the other processing layer. These $4W/s$ wires run to the edge of the cluster slice where they connect to the PE chip in the processing layer (Figure 2.6) or to the CIU in the other communication layer.

The PE chip in the processing layer requires only $2W$ connections to each cluster bar to which it is connected. At each node within the 2-D DCSH array, therefore only $2W$ of the $kW$ wires along a cluster actually exit from a communication layer ($W$ into the PE and $W$ passing to the second dimension). Another $2W$ wires are required to enter the layer. Nonetheless this number is entirely independent of $k$; the "divide down" effect of the CIU multiplexers ensures that the higher wire count within a cluster layer does not appear at its surface.

The wire count in a cluster bar is much higher than that achievable in a purely two-dimensional wiring density system. This can be quantified as follows. Suppose that in a given technology a certain wiring density per centimetre is achievable. Suppose that a bar of s slices is just wide enough to span a PE chip, $b$ cm across, and is $h$ cm deep (Figure 2.8). The DCSH wire count is then

$$2s \times \frac{h}{b} \tag{2.2}$$

greater than a 2-D wiring density technology such as PCB would allow. In practice s is limited by the thickness of a hybrid slice (and the interspacing required between slices to accommodate CIU chips and possible coolant channels); however for $b$ of say 1.5-2 cm, an $s$ of about 8 seems reasonable. In a 2-D system $h$ may be larger than $b$, but it is also limited by the size of the CIU slice chips.

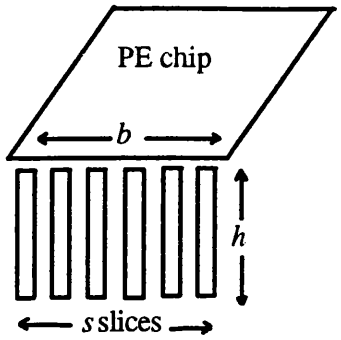



**Figure 2.8: Wire count in the layered implementation**

One remaining implementation issue is that of the long wires inherent in the topology and the effect these might have on signal delay and transmission rates. A typical future implementation might require signals to travel up to 0.3 m at rates of $10^8$-$10^9$ Hz, over wires of diameter as low as 20 μm: while this is a substantial demand, it is not outwith the projected capabilities of hybrid technologies to construct transmission lines with these characteristics [116]. It is clearly undesirable to force clock synchronisation across the whole network if very high clock speeds are being employed, since the clock speed will be limited by the longest path length. Similarly, the use of self-timed [117] circuits will reduce transmission rates to the rate determined by the maximal bit delay. One solution to this problem is to synchronise message paths to the source, using a transmitted strobe: when data is routed through a switching node, the strobe takes the same route, eventually clocking the data into the destination. No handshaking is necessary, and transmission proceeds at the maximum rate of the sender [50, 51, 62].

## Comparison with other networks

In a $k \times k$ DCSH system each channel is $W$ wires wide. The bisection width of the cluster is thus $kW$ wires which is likely to be high for any substantial values of $k$ and $W$. High wiring densities are available inter-chip using current and projected hybrid circuit technology [118]. Of course, such wiring densities are also available to other topologies, and an elementary fixed wiring density argument indicates other hypergraph and graph networks could exploit this to create wider channels. While the DCSH clearly maps well onto the layered structure outlined above, an obvious question arises as to whether any other topology can actually take advantage of a similar means of increasing available wiring density. The advantage of the DCSH is that it requires far lower wire counts at the interface to a PE or the interface between dimensions (critical relay bandwidth) than within a cluster. This means that the number of wires which have to "surface" at a PE is only a fraction, $2/k$ of the number of wires travelling along a cluster. In contrast,

spanning-bus hypercubes for example have no equivalent "multiplexing down" function between dimensions, so a SE must be able to relay the entire bandwidth of a row and column and vice-versa. Were a cluster bar implementation technology to be employed, all wires would have to surface to all nodes, imposing a huge bandwidth density requirement, and thereby nullifying the advantage of layering.

Meshes, tori, binary $n$-cubes, and generalised hypercube encounter the same relay bandwidth problem [56]. The inherent non-uniformity of flattened binary $n$-cubes and generalised hypercubes would in any case make a sliced and layered implementation a complex and asymmetrical undertaking.

### 2.4.2 DCSH variations

In the above implementation scheme, it is clearly envisaged that a DCSH cluster would be laid out in a single physical dimension. An enhancement is therefore possible as follows. In order to take full advantage of the channel bandwidth, the channel can be split into two, one running left from the node, the other right. This has no deleterious effect on overall wiring density, although there is a small additional cost in buffering internal data paths. This structure will be referred to as an *asymmetric-split*-DCSH (AS-DCSH), because the two channels leaving a node are, in general, unequal in length.

It is possible to remove this asymmetry using channel wrap-around in the sliced cluster bar implementation so that two channels from a node are of equal (or almost equal) length. Such a structure will be called a *symmetric-split*-DCSH (SS-DCSH), and has the same pin-out requirement as an AS-DCSH, but its wiring density is higher.

The idea can be generalised to increase the number of channels from a node to $C$, where each is used to communicate with $k/C$ nodes of the cluster. If $C=k$-1 the structure obtained is the generalised hypercube.

## 2.5 Multi-stage interconnection networks (MINs)

One network class which cannot easily be classified as graphs and hypergraphs are MINs, which have been mainly used in shared-memory multiprocessors [84, 85, 86]. They have been suggested as an attempt to strike a balance between the implementation cost and performance alternatives offered by the shared bus and crossbar switch. An $N\times N$ MIN connects the $N$ nodes to one another by employing multiple-stages of banks of

crossbar switches of small sizes. Several MINs have been proposed [85, 86]. Although larger systems are possible than with a single shared bus and crossbar switch, they have been precluded from use in multicomputers because their cost becomes the dominant factor in large systems and they are unable to exploit any degree of communication locality [52, 119, 120].

## 2.6 Conclusions

Hypermeshes, which are regular multi-dimensional hypergraph networks, have significant advantages over traditional graph networks: they possess diameters which grow slowly with number of nodes: many common structures such as binary trees, hypercubes, meshes of two, three or more dimensions, can be efficiently embedded in such structures; they emulate some multi-stage interconnection networks such as Omega and Inverse Omega; they efficiently perform broadcast operations such one-to-all and all-to-all. Low-dimensional distributed crossbar switch hypermeshes (DCSHs) in particular lend themselves to efficient implementation. Demand is likely to grow in the future for systems which provide efficient support for both local and non-local communications, and again, hypermeshes are ideal candidates.

The aim of the following chapters is to demonstrate quantitatively that DCSHs, low-dimensional ones in particular, deserve serious consideration as high-performance interconnection networks in future multicomputers. Their performance will be analysed and compared with other important networks and demonstrate that they constitutes an attractive alternative even when other implementation schemes, other than the layered implementation, are used, notably VLSI and multiple-chip technology.

This endeavour is divided into four main parts. Firstly, DCSH queueing models are derived and verified against simulations to examine both local and non-local communication. These models are used to assess the relative performance of the most common switching methods, routing algorithms, and different switching element designs. Secondly, the DCSH is compared to hypermesh implementations using shared-bus, crossbar switch, and complete-connection. Thirdly, the DCSH is compared to the most common graph network, namely the mesh. Finally, the performance merits of the DCSH are assessed against the multi-stage interconnection network, and since only low-dimensional DCSHs fit naturally with the layered implementation, they are evaluated against their higher-dimensional counterparts to examine whether they have superior performance, when other technologies are used.

# 3 Performance of distributed crossbar switch hypermeshes

## 3.1 Introduction

Interconnection networks performance is usually measured by two common metrics, notably *latency*, the time required for a message to cross from source to destination, and *throughput*, the number of messages successfully delivered to destinations. An optimal network should have minimum latency and maximum throughput.

This chapter examines the performance characteristics of multi-dimensional DCSHs. It assesses the impact of the switching methods and routing algorithms on their performance. Moreover, it evaluates alternative switching element (SE) designs and variations of the basic DCSH topology. Queueing models are used in the following analyses whenever they are possible. Since simplifying assumptions are often made to reduce the complexity of the queueing models, there is a need to validate the models through simulation. This validation is carried out for small test cases which require reasonable computation time and resources. The significant advantage of the analytical approach over simulation is that the queueing models can be used to obtain performance results for large systems which are infeasible by simulation due to the excessive computation demands on conventional computers.

## 3.2 Analysis of switching methods

For the purpose of the following discussions, it is assumed that message routing is restricted because this routing strategy makes a high performance SE simple to implement [4, 65, 66, 76]. A node generates a message of fixed length ($B$ phits) with probability $m$ in a cycle (a cycle is the time taken to transmit a phit) and has a destination chosen from the remaining ($N$-1) nodes with equal probability (i.e., uniform reference model). In the following sections, analyses of virtual cut-through (VCT), message switching (MS), and wormhole routing (WR) are presented. Circuit switching (CS) is omitted because simulation results have revealed that it has the worst performance [103, 121].

### 3.2.1 Virtual cut-through model

Message queues are assumed of infinite capacity. Simulation experiments show that the assumption of infinite buffering is realistic [103, 122, 52, 54]. Under the uniform reference model, as few as three or four message buffers at each queue can provide performance approaching that for infinite buffers. The network will operate in traffic regions where channel utilisation is moderate (<80%), so only a few buffers are ever occupied. The performance regions close to 100% utilisation should be avoided in any case as network transit time becomes extremely long. Message latency in the absence of blocking is easy to compute, but it cannot predict performance adequately [51]. Realistic finite-buffer analyses are generally computationally intensive, and infinite-buffer analyses are a reasonable compromise [54].

Figure 3.1 shows the SE structure in the 2-D DCSH. However, the discussion can be easily extended to an $n$-dimensional DCSH network. Let dimensions be numbered from 1 to $n$ with messages visiting higher-numbered dimensions first. Figure 3.1 shows that due to restricted routing, only messages from the local PE may access dimension 2. Furthermore, only messages from the local PE and dimension 2 may access dimension 1. However, messages can be transferred to the local PE from any dimension.

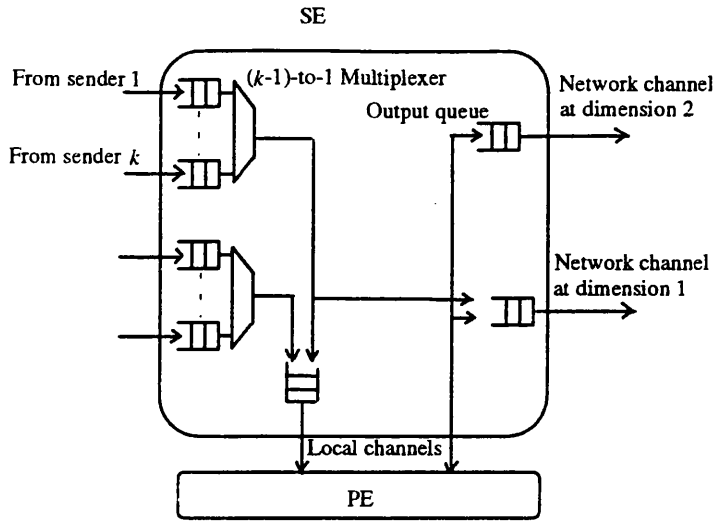



**Figure 3.1: The SE structure in the 2-D DCSH using virtual cut-through.**

Buffers are provided at the input and output sides of the SE. Messages generated from the PE are copied into one of the $n$ output queues associated with the required network channel. There is also one $(k\text{-}1)$-to-1 multiplexer per dimension. In each entry of multiplexer $i$ $(1 \leq i \leq n)$, there is an input queue for node $j$ $(1 \leq j \leq n,\ j \neq i)$ at dimension $i$. Each input queue consists of message buffers each of which has enough storage to hold temporarily an entire message (header plus data phits) if the required channel is busy. When a PE generates a message, a routing decision is made to select the output queue into which the message is copied. When a message arrives at the multiplexer for dimension $i$, its header phit is buffered. If multiplexer $i$ is free, a routing decision is made to select the next channel. It is then copied into an output queue associated with one of the network channels or the local channel leading to the PE if it is destined to that PE. A message can bypass an output queue if it is empty.

Following the terminology of Abraham, the above SE structure, which has a single local channel to the PE, will be referred to as the *single-accepting* model (SAM) [28, 34]. With increased implementation cost, the SE can have multiple parallel local channels leading directly to the PE; different messages, arriving along different dimensions, can be simultaneously transferred to the PE. This SE structure will be referred to as the *multiple-accepting* model (MAM) [28, 34]. It has been used in the recent multicomputer the iWARP [45]. The following discussion developed a queueing model using SAM initially and then shows how it can be easily adapted for MAM.

The model is derived first for unit-length messages and zero decision times. It is then

extended to include the effects of larger messages and decision times. The model uses the following assumptions.

a) The SE operations are synchronous; an SE sends and receives all messages at the beginning and end of a cycle respectively.

b) Messages generated from the local PE are steered to network channels with equal probability.

c) Output queue $i$ ($1 \le i \le n$) can receive messages from any multiplexer $j$ ($j>i$) and the PE, and therefore these messages may block each other. However, this analysis assumes that at an output queue $i$, all message blocking is ignored except that which occurs between messages that originated from the PE and those which switch from dimension $i$+1 to $i$.

Assumption (a) simplifies the analysis because it makes the different probabilities easier to compute [28, 34, 54, 122], Assumptions (b) and (c) model the effects of restricted routing while keeping the analysis simple [52].

Under the uniform reference model, the average number of visited SEs excluding the destination, $d$, in a DCSH is the same as in the generalised hypercube and is given by [109, 110]

$$d = n\frac{(k-1)}{k}\frac{N}{(N-1)} \tag{3.1}$$

where the factor $N/(N\text{-}1)$ accounts for the fact that nodes do not reference themselves.

The probability that a PE generates a message in a cycle is $m$. The probability that there is a message on a network channel in a cycle, $\rho$, using the assumption that messages from the PE are steered to any one of the $n$ network channels with equal probability and visit on average $d$ SEs, is given by

$$\rho = \frac{md}{n} \tag{3.2}$$

Given the definition of unit-sized messages, this is also the channel utilisation.

Since cluster nodes are referenced equally probably, the probability that in any cycle a sender node references one of ($k$-1) destinations is $\rho/(k-1)$. This is also the traffic rate

at a given input multiplexer entry. The ($k$-1) queues in a multiplexer can be treated as a single queue with ($k$-1) input streams of rates $r_j=\rho/(k-1)$, $(1 \le j \le k-1)$. To determine the mean waiting time at a multiplexer, the multiplexer queue is modelled as a G/D/1 queueing system. The mean waiting time in a G/D/1 queue, with $k$ input streams of rates $r_1, r_2, ..., r_k$, is given by [123, 124]

$$w = \frac{V}{2E(1-E)} - \frac{1}{2} \tag{3.3}$$

where $E$ is the expectation of the arrival process and $V$ is the variance of the arrival process. $E$ and $V$ are given by the following equations [123, 124]

$$E = \sum r_j \tag{3.4}$$

$$V = \sum r_j(1-r_j) \tag{3.5}$$

Equation 3.2 gives the average waiting time, $w_m$, at a multiplexer as

$$w_m = \frac{\rho}{2(1-\rho)} \frac{(k-2)}{(k-1)} \tag{3.6}$$

When a message is either generated by the PE or selected by a multiplexer, if it has not yet reached its destination it is switched to an output queue associated with the required network channel. Otherwise, it is copied into the output queue associated with the local channel. The average waiting time at an output queue depends on the rate of messages that switch from one dimension to another. The probability ($\rho$) that in a given cycle there is a message on a network channel at dimension $i$ $(1 \le i \le n)$ is composed of two components.

$\rho_t$: the probability that the PE injects a message into dimension $i$. In the steady state, $\rho_t$ is also the probability that a message exits the network from dimension $i$.

$\rho_s$: the probability that a message switches from dimension $i$+1 to $i$.

The probability that a message is generated by the PE in a cycle is $m$, and the probability that this message routes to a given network channel is $1/n$, yielding

$$\rho_t = \frac{m}{n} \tag{3.7}$$

Since the probability that a message exits the network from a dimension is $\rho_t$, the probability that it stays in the network is $(\rho - \rho_t)$. If a message does not exit the network,

it can switch only to the next lower dimension (assumption c), and therefore the probability that it does so is given by

$$\rho_s = (\rho - \rho_t) \tag{3.8}$$

$\rho_t$ and $\rho_s$ can be rewritten using the terms defined by Abraham [34, 28]. Under the uniform reference model, a message visits on average $d$ SEs excluding the destination. A message which arrives at an SE with hop count=$d$ is destined for the local PE. In the steady state, the expected number of messages with hop count=$i$ will be the same for all $i$, $(1 \leq i \leq d)$. Thus the probability of termination, which is the probability that a message arriving at an SE is destined for its corresponding PE, can be written as [34, 28]

$$P_t = \frac{1}{d} \tag{3.9}$$

In term of $P_t$, $\rho_t$ and $\rho_s$ are given by

$$\rho_t = P_t \rho \tag{3.10}$$

$$\rho_s = (1 - P_t)\rho \tag{3.11}$$

The output queue associated with a network channel can also be modelled as a G/D/1 queue with more than one input stream. There are two input streams to this queue: one from multiplexers $i$+1 with rate $\rho_s$ and another from the PE with rate $\rho_t$. Equation 3.3 yields the mean waiting time at this queue as

$$w_o = \frac{(1 - P_t)P_t\rho}{(1 - \rho)} \tag{3.12}$$

Similarly, the output queue associated with the local channel has $n$ input streams each of which has a rate of $\rho_t$. The mean waiting time at this queue is

$$w_l = \frac{(n-1)P_t\rho}{2(1 - nP_t\rho)} \tag{3.13}$$

The model can be extended to include non-unit-sized messages. When each generated message has $B$ phits, the channel utilisation increases correspondingly by a factor $B$, yielding [28, 34, 54, 122]

$$\rho' = \rho B \tag{3.14}$$

In a given queue, the $i^{th}$ message must wait $(i\text{-}1)B$ cycles for its turn to be serviced. Therefore, the waiting times through the different queues must by scaled by a factor $B$ to

reflect the increase in message service time [28, 34, 54, 122]. Equations 3.6, 3.12, and 3.13 then become

$$w'_m = \frac{\rho' B}{2(1-\rho')} \frac{(k-2)}{(k-1)} \tag{3.15}$$

$$w'_o = \frac{(1-P_t)P_t\rho' B}{(1-\rho')} \tag{3.16}$$

$$w'_l = \frac{(n-1)P_t\rho' B}{2(1-nP_t\rho')} \tag{3.17}$$

The above waiting times account only for the blocking that a message header encounters at an SE. The delay due the transmission time of the $B$ flits has also to be included. A message crosses on average $d$ SEs before it reaches its destination. Further, it requires an extra cycle each time it moves from one queue to the next. Taking into account the mean waiting times in all the different queues gives the mean latency as

$$L' = (1+w'_o)d + (1+w'_m)d + (1+w'_l) + B \tag{3.18}$$

When a message header reaches an intermediate SE, a routing decision selects the next channel. As realistic values for the decision time (e.g., 3 cycles) are far smaller than message transmission times, (which are considered throughout the next sections and chapters) simulations reveal that its effect on message latency is negligible at high traffic load [2, 4, 65, 66, 76]. Therefore, it is reasonable to consider only the effect of the decision time under light traffic conditions. The mean total latency becomes

$$L'' = (1+w'_o)d + (1+w'_m)d + (1+w'_l) + (D_t+1)d + B \tag{3.19}$$

With MAM, messages from different multiplexers are transferred directly to the local PE. Therefore, the total mean latency can be obtained by simply excluding the third term of Equation 3.18 which becomes

$$L''' = (1+w'_o)d + (1+w'_m)d + (D_t+1)d + B \tag{3.20}$$

## Model validation

The above model has been validated by means of a discrete-event simulation model (or simulator for short) [125]. The program code was written in the simulation programming language Simscript II.5 [126]. A network size of 256 nodes was used due to the large computing cost that is needed when running the simulator for large network sizes. The

following assumptions have been made in the simulator.

a) Each node generates messages independently of any other node, with probability $m$ in a cycle (a cycle is the time to transmit a phit).

b) Messages are of fixed length.

c) Message destinations are uniformly distributed across the network

d) Routing is restricted: the simulator routed messages through the network in the order of decreasing dimensions.

e) The switching method is VCT.

In addition the model has been run with the following variable parameters.

f) Each input in the multiplexer can hold four messages simultaneously.

g) Each output queue can hold four messages simultaneously.

k) Each SE has a zero decision time.

In order to gather statistics about the behaviour of the network, a generated message has some associated information.

- T_Create: the time when a source PE generates a message.
- T_Arrive: the time when the destination PE receives the last phit of a message.
- T_Send: the time when a given queue services a message.
- T__Receive: the time when a given queue receives a message

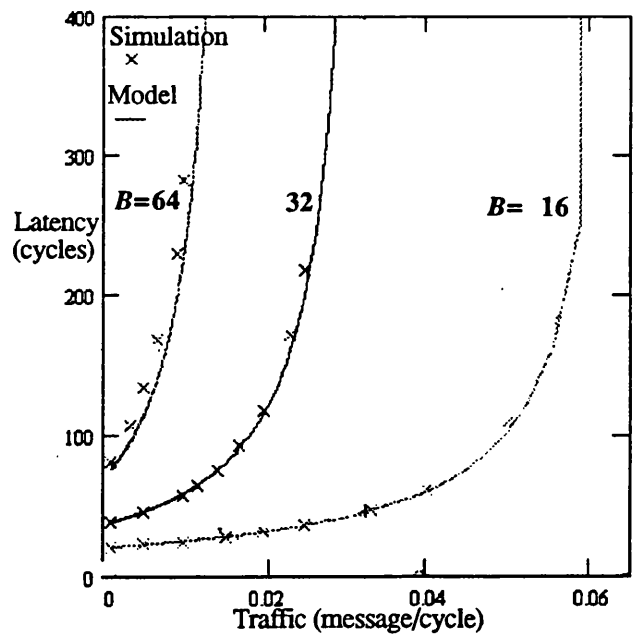
These quantities are used to determine:

- Average_Total_Latency: the time between a source PE generating a message and the destination PE receiving the last phit of the message.
- Average_Waiting_Time: the time spent by a message in a given queue before receiving service.
- Average_SE_Time: the time spent by a message at a given SE.

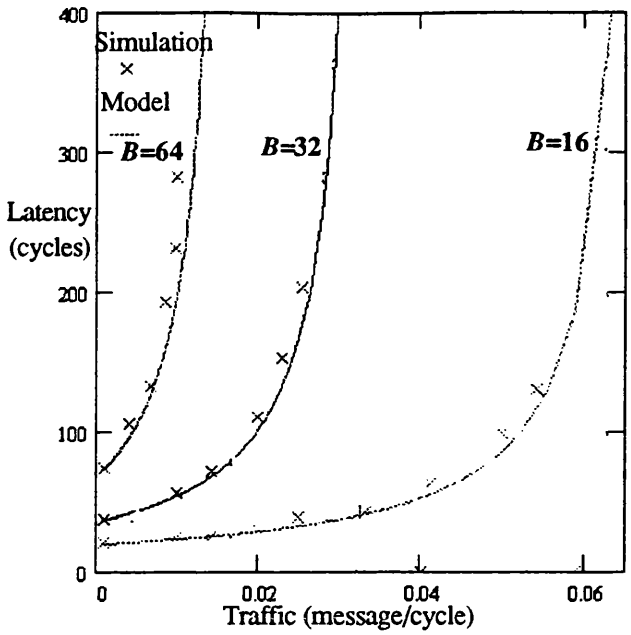
Each simulation experiment is run until the network reaches steady state, that is until a further increase in simulated network cycles does not change the collected statistics appreciably.

Figures 3.2 and 3.3 compare the latency predicted by the above queueing model against the simulator. Results are shown for $16^2$ and $4^4$ node systems with zero decision time. Results for both SAM and MAM and different message lengths ($B$=16, 32, and 64 phits) are presented. The horizontal axis represent the number of messages generated by a node in a cycle, and the vertical axis represents the mean latency in cycles.

The mathematical model slightly underestimates latency because of the synchronised-SE assumption. The figures confirm that considering only contention at an output queue in dimension $i$ between messages generated from a PE and those originating from dimension $i$+1 yields results that are reasonably accurate. Furthermore, simulations show that two message buffers at each input to a multiplexer together with four buffers at an output queue are enough to approach the performance of the infinite-message-buffers assumption. The performance predictions of the model yield results that are reasonably accurate up to a channel utilisation of 80%. It can, therefore, be concluded that the errors introduced by the queueing model assumptions are generally negligible.

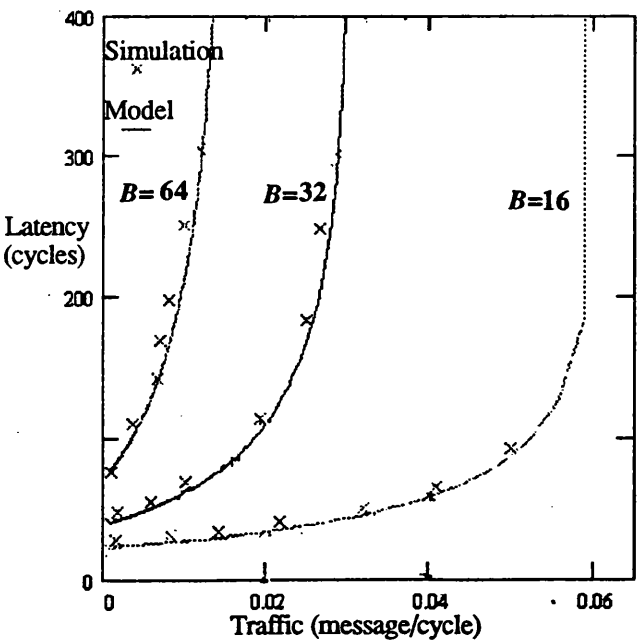



**a) SAM**

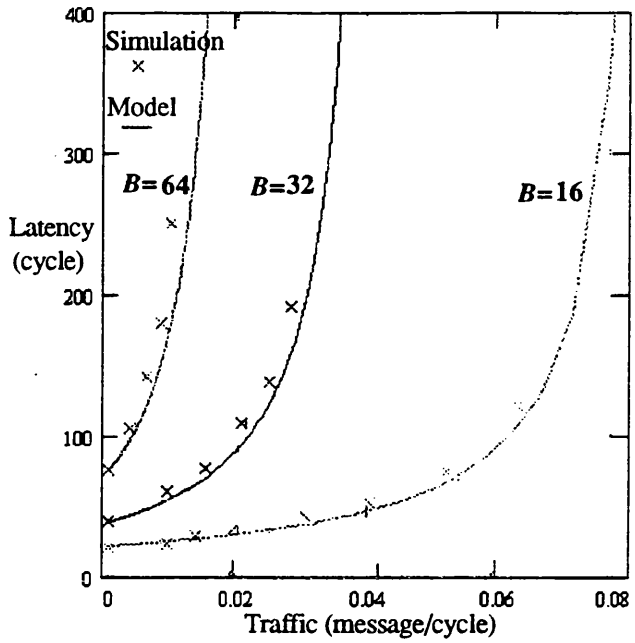



b) MAM

**Figure 3.2: Comparing the virtual cut-through model with simulation. $n$=2, $N$=256.**




a) SAM




b) MAM

**Figure 3.3: Comparing the virtual cut-through model with simulation. $n$=4, $N$=256.**

As the message length increases, the network saturation point decreases, a behaviour which can be explained by the expression $\rho = mBd / n = 1$. The saturation value $m_s$ varies directly as $1/B$. Further, as the dimensionality increases, the traffic rate on network channels decreases, and therefore latency is reduced. Finally, when the network width ($k$) is varied, the network saturation point remains unchanged as it has a negligible effect on the average message distance, and therefore on the channel traffic rate.

### 3.2.2 Message switching model

The mean latency for MS is obtained by a simply modifying the VCT model to account for the message buffering that occurs at each queue. A message visits on average $d$ output queues and multiplexers. Furthermore, with SAM, it is also buffered at the output queue associated with the local channel. Therefore, for an SE with a zero decision time, the mean total latency is given by

$$L = (1 + w_o')d + (1 + w_m')d + (1 + w_l') + B(2d + 1) \qquad (3.21)$$

where $w_o$, $w_m$, and $w_l$ are given by Equations 3.15, 3.16, and 3.17 respectively. With MAM, the latency is obtained by simply ignoring the third term in Equation 3.20.

Under light traffic, MS suffers from an extra $2Bd$ cycles due to buffering overhead compared to VCT. Under high traffic, the same conclusions (e.g., the saturation rate) are reached as with VCT.

### 3.2.3 Wormhole routing model

For the sake of simplicity, this analysis assumes that traffic generated from each PE follows an independent Poisson process and also uses MAM for the SE [22, 37] (simulation results have shown that with SAM, the message latency does not change much since the bottleneck does not reside at the channel leading to the local PE).

As Figure 3.4 shows, buffers are provided only at the input side of the SE because this is suitable for the pipelined nature of WR. Messages generated by the PE are buffered in a FIFO queue to be injected later into the network. There are also $n$ ($k$-1)-to-1 multiplexers, one for each dimension. At each input of a multiplexer, there is a queue for each of the ($k$-1) senders in a dimension. Each queue can hold only few flits[3] (a header flit plus possibly some data flits). When a message is selected for transmission, a routing decision selects the next channel. As soon as the message is granted the channel, the header flit resumes transmission along with the data phits which arrive in a pipelined fashion. Messages destined for the PE can be transferred simultaneously through multiple local channels to the PE.

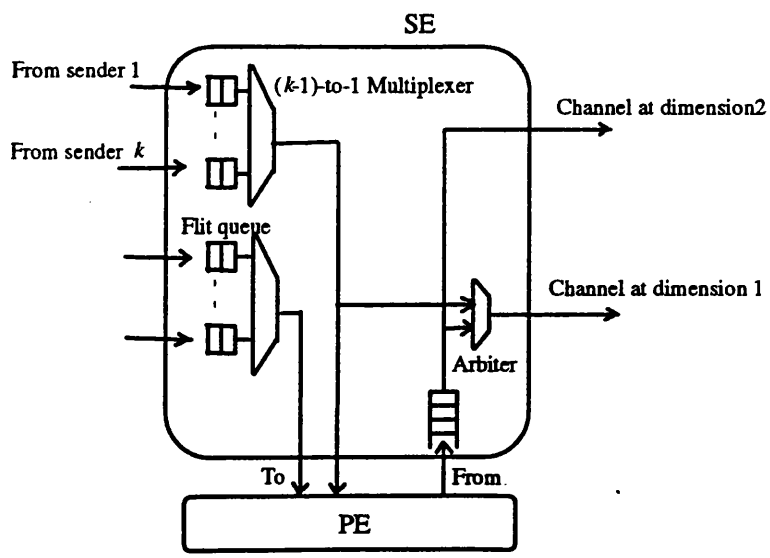



**Figure 3.4: The SE structure in the 2-D DCSH using wormhole routing.**

[3] Recall that a flit is typically composed of a few phits.

Under very light traffic ($m \approx 0$), there is hardly any message blocking in the network, and the message latency is simply given by

$$L = (d+1) + B \tag{3.22}$$

The first term accounts for the number of hops that a message has to make to reach the destination PE, while the second accounts for the message transmission time (the "+1" accounts for the last hop at the destination SE to reach the local PE).

As traffic increases, a message may be spread across several network channels at the same time. Thus, the message service time at any dimension is dependent on that at the succeeding dimensions. All dimensions have therefore to be considered in the calculation of message latency. As in [22, 37], the network is partitioned into dimensions and latency is calculated in each dimension separately, adding the extra latency due to message routing that occurs in the lower dimensions. Let dimensions be numbered from 1 to $n$; a message visits higher-numbered dimensions first with the destination node at the fictive dimension 0. Since the restricted routing assumption is maintained, once a message travels to a lower dimension, it never again visits a higher one.

Latency seen entering dimension 0 (i.e., the destination) can be computed and then propagated back to dimension $n$ (i.e., the source). Since flits are serviced as they arrive at the destination, the latency is given by

$$L_0 = B \tag{3.23}$$

Figure 3.5 shows the model to calculate the latency seen entering a dimension. When messages enter a dimension, a fraction $\alpha = 1/k$, of them are already at the right co-ordinate in that dimension. These messages skip over the dimension. Another fraction $(1-\alpha)$ have to pass through dimension to correct the appropriate co-ordinate corresponding to that dimension. The traffic injected from the PE into the network is $m$. Recalling that nodes do not reference themselves, the total traffic arriving at dimension $i$ from the previous dimension is given by

$$m' = \frac{N}{(N-1)} m \tag{3.24}$$

As stated before, this traffic is composed of two components: a component that skipped dimension $i$+1, and a component that passed through dimension $i$+1. Their rates are given by the following equations respectively (the subscript $p$=pass, $s$=skip)

$$m_p = (1-\alpha)m' \tag{3.25}$$

$$m_s = \alpha m' \tag{3.26}$$

The components of these two streams that pass through dimension $i$ are (the first subscript for dimension $i$+1 and the second for $i$)

$$m_{pp} = (1-\alpha)^2 m' \tag{3.27}$$

$$m_{sp} = \alpha(1-\alpha)m' \tag{3.28}$$

These compete for access to the network channel corresponding to dimension $i$. Similarly, the components of the two streams that skip dimension $i$ are

$$m_{ps} = \alpha(1-\alpha)m' \tag{3.29}$$

$$m_{ss} = \alpha^2 m' \tag{3.30}$$

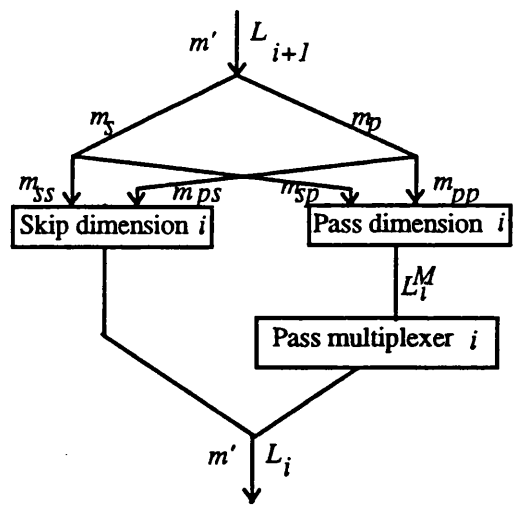



**Figure 3.5: A model to determine latency in a DCSH dimension.**

These must compete for a network channel when they re-enter another lower dimension. Let $L_i$ be the latency seen by a message entering dimension $i$. Figure 3.5 shows that only messages which pass through dimension $i$ encounter blocking at multiplexer $i$. Message blocking occurs, when more than one message arrive at multiplexer $i$. The increase in latency due to the blocking is given by [22, 37]

$$L_i^M = P(\textit{Message blocking})E(\textit{Message delay}) \tag{3.31}$$

where *P(Message blocking)* is the probability that the message blocking occurs and *E(Message delay)* is the expected message service time. Since $m_p$ messages pass through dimension $i$, the traffic rate arriving at a multiplexer entry is $m_p/(k-1)$. A message therefore sees on average

$$m_m = \frac{(k-2)m_p}{(k-1)} \tag{3.32}$$

other messages at the multiplexer. Given that the message service time at the multiplexer is $L_i$, the probability that blocking occurs is

$$P(\textit{Message blocking}) = m_m L_i \tag{3.33}$$

If blocking does occur, the waiting time has a uniform distribution from 0 to $L_i$. Thus the expected message delay can be written as

$$E(\textit{Message delay}) = \frac{L_i}{2} \tag{3.34}$$

Therefore, the increase in latency due to message blocking in multiplexer $i$ can be written as

$$L_i^M = \frac{m_m L_i^2}{2} \tag{3.35}$$

The latency seen by a message when entering dimension $i$, taking into account the different traffic components which pass and skip the dimension with their appropriate weights, is given by

$$\begin{aligned} L_{i+1} = L_i + (1-\alpha)L_i^M + \alpha(1-\alpha)^3 m'(L_i + L_i^M)^2 \\ + \alpha^3(1-\alpha)m' L_i^2 \end{aligned} \tag{3.36}$$

As depicted in Figure 3.5, the first term of Equation 3.36 is the latency seen by a message when entering the next lower dimension. The second term accounts for the increased latency due to blocking at multiplexer $i$. The third term accounts for blocking between the two streams that pass through dimension $i$. The final term gives the latency due to blocking between the two streams that skip dimension $i$.

Finally, the effect of queueing which occurs at the source must also be included. It is assumed that messages are serviced at the source according to an exponential distribution with an average service time $L_n$. Simulations have shown that this

assumption gives more accurate delays than if the message service were assumed deterministic. M/M/1 queueing theory then gives the mean latency as [123]

$$L = \frac{L_n}{1 - mL_n} \tag{3.37}$$

Once a message header reaches an SE, a routing decision selects the next channel. Let the decision time be $D_t$ cycles. The message service time at dimension $i$ is therefore increased by a factor $D_t$. Equations 3.21, 3.23, 3.36 become

$$L' = (d+1)D_t + B \tag{3.38}$$

$$L'_0 = B + D_t \tag{3.39}$$

$$L'_{i+1} = D_t + L'_i + (1-\alpha)L_i^M + \alpha(1-\alpha)^3 m'(L'_i + L_i^M)^2 + \alpha^3(1-\alpha)m'L_i^{2} \tag{3.40}$$

## Model validation

A Simscript simulator has also been built to verify this queueing model. The simulator uses almost the same assumptions as that of Section 3.2.1 except that

e') The switching method is WR.

f') Each input in the multiplexer can hold two flits simultaneously. This assumption enables full exploitation of the pipelining nature of WR. It enables an SE to make a look-ahead acknowledgement [51, 62]; as soon as an intermediate SE stars emptying its FIFO flit buffer, it informs the previous SE that a flit slot is available. By the time it has finished transmission, new flits have arrived and their transmission can begin immediately. This cannot be done when only a single buffer is used to store flits because extra cycles have to be introduced during transmission. This means that in a given cycle, leaving and incoming flits can be sharing the same FIFO queue.

k') Although 0-cycle decision time is unrealistic, decision times assumed 0 and 3 cycles.

Figures 3.6 and Figure 3.7 compare network latency predicted by the queueing model against the simulator for the 2-D and 4-D DCSH with $N$=256 and zero decision time. Figures 3.9 and 3.10 compare results from both models when the decision time is increased to 3 cycles.

The queueing model yields delay results which are close (within 8% error) to those provided by the simulation model under light and moderate traffic. However, the difference in latency between the two models gets larger as traffic becomes heavy. Since the errors introduced by the queueing model in the traffic regions of interest are reasonably small, it will nevertheless serve our purpose.

Both the simulation experiments and the queueing model show that in WR, the bottleneck resides at the source: while the network transit time $L_n$ is always finite, the queue which receives messages from the local PE can blow up to infinity. This occurs when PEs inject more messages than the network can deliver to destinations. Examining Equation 3.19 shows that the PE queue saturates when $m = 1/L_n$. There are different parameters which can influence the network saturation point. When the message length increases, the network saturation point decreases since $L_n$ has a polynomial relationship with $B$. Further, as the dimensionality increases, the network saturation point decreases. This is because the number of blocking stages increases with $n$, therefore causing $L_n$ to increase. The network width, on the other hand, does not affect the network saturation point as it has only insignificant influence on $L_n$.

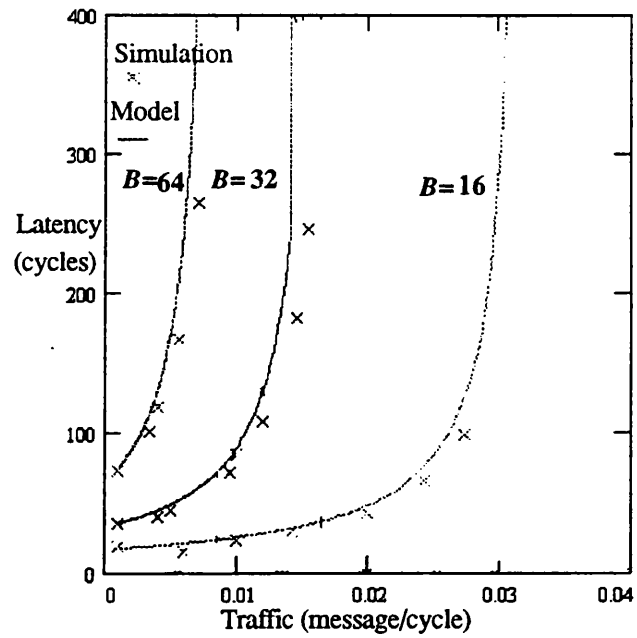



**Figure 3.6: Comparing the wormhole routing model with simulation.**
**$n$=2, $N$=256, $D_f$=0.**

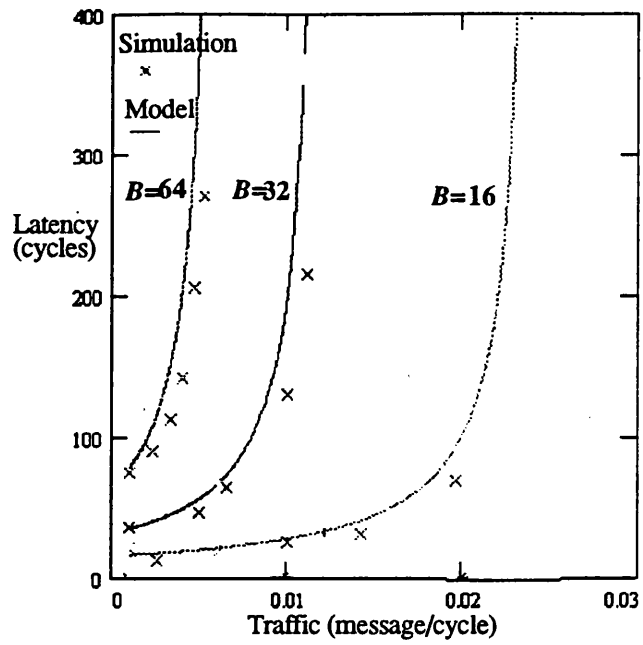



**Figure 3.7: Comparing the wormhole routing model with simulation.**
**$n$=4, $N$=256, $D_f$=0.**

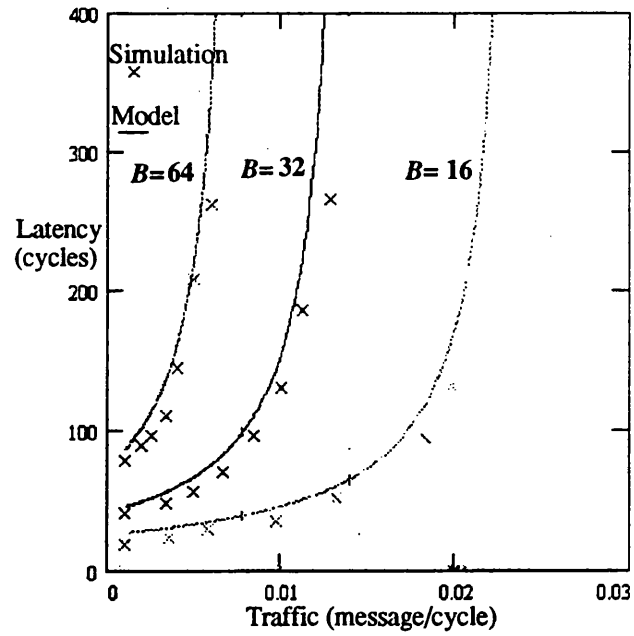



**Figure 3.8: Comparing the wormhole routing model with simulation.**
**$n$=2, $N$=256, $D_f$=3.**

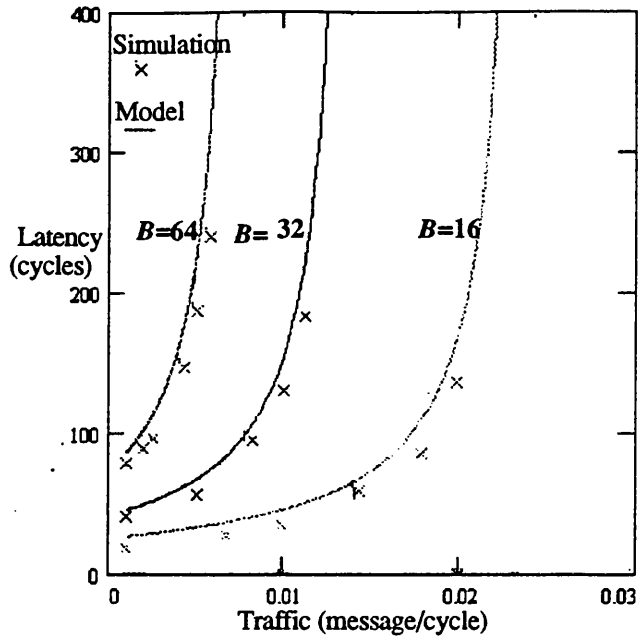



**Figure 3.9: Comparing the wormhole routing model with simulation.**
**$n$=4, $N$=256, $D_f$=3.**

### 3.2.4 Comparison of switching methods

The queueing models of the previous sections may be used to assess the relative performance merits of VCT, MS, and WR in the 2-and 4-D DCSH. The results are presented in Figures 3.10 and 3.11 respectively where MAM is used, $N$=4096 nodes, and $B$=128 bits (Results for SAM and other network and message sizes yield the same conclusions).

Figure 3.10, which shows message delays for the 2-D DCSH, suggests that at light traffic VCT and WR have the same performance since in both methods a message has a high chance of cutting-through intermediate SEs. In VCT and WR, latency is some 70% better than in MS because of the whole-message buffering overhead in the latter. At fairly high traffic, VCT and MS have considerably lower latency than WR. In WR, instead of phits being buffered when message blocking occurs, as in VCT and MS, they remain spread across network channels resulting in higher delays. Under high traffic VCT degenerates to MS as the opportunity of any cut-through vanishes. The figure indicates that there is a traffic load where MS and WR have the same performance and this occurs at approximately $m$=0.0036. At this point the gain obtained from avoiding the buffering overhead is offset by message blocking over network channels.

Figure 3.11 shows that as the dimensionality increases ($n$=4), the relative merits of VCT and WR over MS are even greater under light traffic. WR runs into trouble much faster than in the 2-D DCSH due to the increase in the number of blocking stages. VCT stays almost unchanged. MS suffers but nearly so much as WR. The crossover between MS and WR occurs at $m$=0.0023, a difference of nearly 13 from the 2-D case.

As the dimensionality further increases, the merits of WR are reduced as its performance degrades considerably. MS also becomes less attractive due to the increase in latency under light traffic. VCT is the optimal switching method to use if the complexity of SEs can be realised. MS reduces the SE complexity significantly compared to WR and VCT. Lower-dimensional DCSHs can use MS, and therefore use cheaper SEs, without incurring significant performance loss compared to higher-dimensional structures. Further, MS becomes more cost-effective in lower-dimensional DCSHs especially if messages are short as the buffering overhead is reduced.

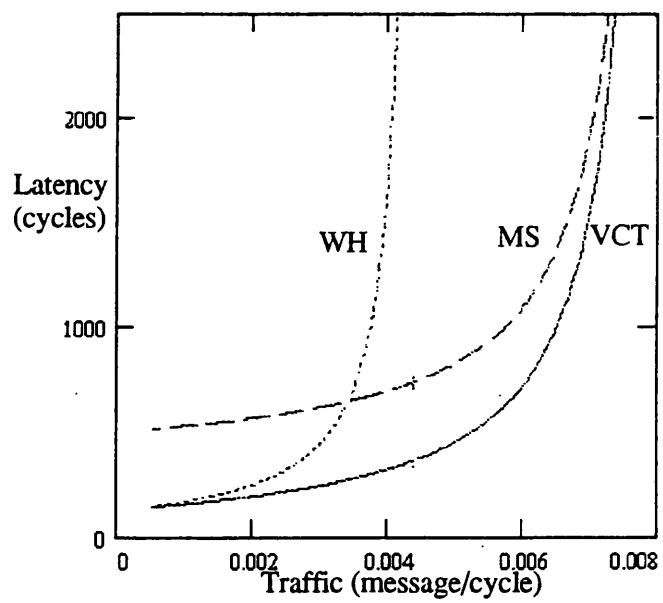



**Figure 3.10: Performance of the switching methods.**

***n*=2, *N*=4096.**

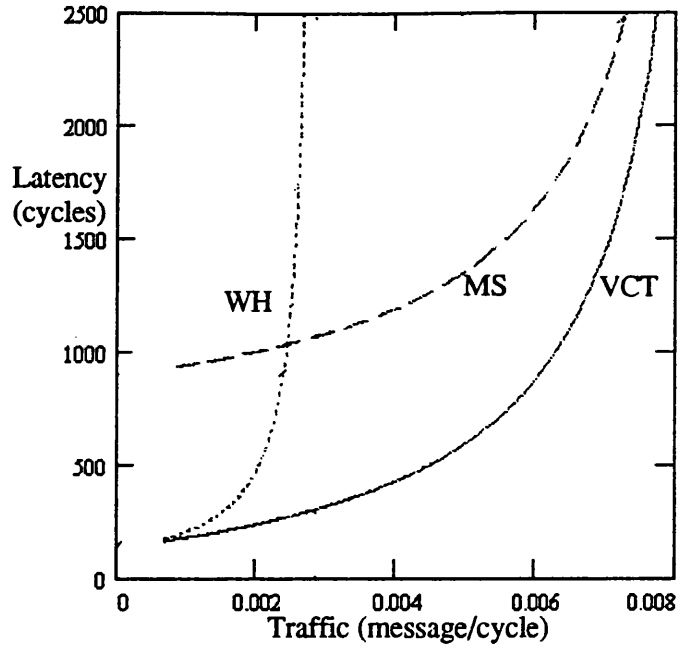



**Figure 3.11: Performance of the switching methods.**
***n*=4, *N*=4096.**

## 3.3 Analysis of routing algorithms

As restricted routing has already been considered in the previous section, the following discussion deals with random routing.

### 3.3.1 Random routing

When a message arrives at an intermediate SE the set of shortest paths leading to the destination are determined. One of the network channels corresponding to a given path is selected at random.

With VCT, the SE structure is similar to that for restricted routing except that any multiplexer can access any output channel. With WR, when random routing is used virtual channels algorithms must be used to avoid message deadlock [81, 82]. These algorithms require extra buffering resources. WR is excluded from discussion in this section since adding extra buffering resources is bound to improve performance as message traffic rates to the different message queues decrease.

Only assumption (a) of Section 3.2.1 is used here (assumptions (b) and (c) were mainly used to simplify the analysis of restricted routing). The following analysis is presented

only for multiple phit messages and zero decision times. The probability that a local PE generates a message in a cycle is $m$. The message must travel on average $d$ hops. Because each SE has $n$ associated network channels, the probability that there is a message on a network channel in a cycle is given by

$$\rho = \frac{m}{n} dB \tag{3.41}$$

The expression for the mean waiting time at a multiplexer is the same as that given by Equation 3.15.

In random routing, since any multiplexer can access any output queue associated with a network channel, the different switching probabilities are

$$\rho_t = P_t \rho \tag{3.42}$$

$$\rho_s = \frac{(1-P_t)\rho}{(n-1)} \tag{3.43}$$

Using Equation 3.3 gives the mean waiting time at an output queue associated with a network channel as

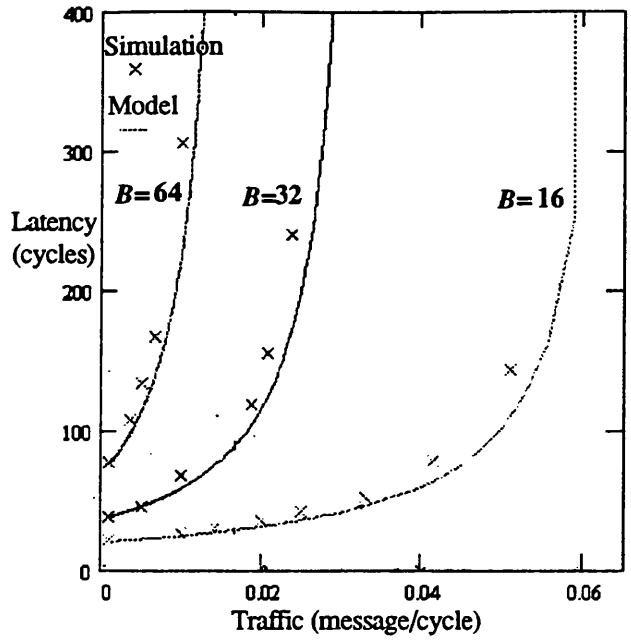
$$w_o = \frac{\rho B}{2(1-\rho)} \frac{(1-P_t)(nP_t + n - 2)}{(n-1)} \tag{3.44}$$

With SAM, the expression of the mean waiting time at the output queue associated with the local channel is the same as that given by Equation 3.17. (With MAM, on the other hand, this mean waiting time is not included).

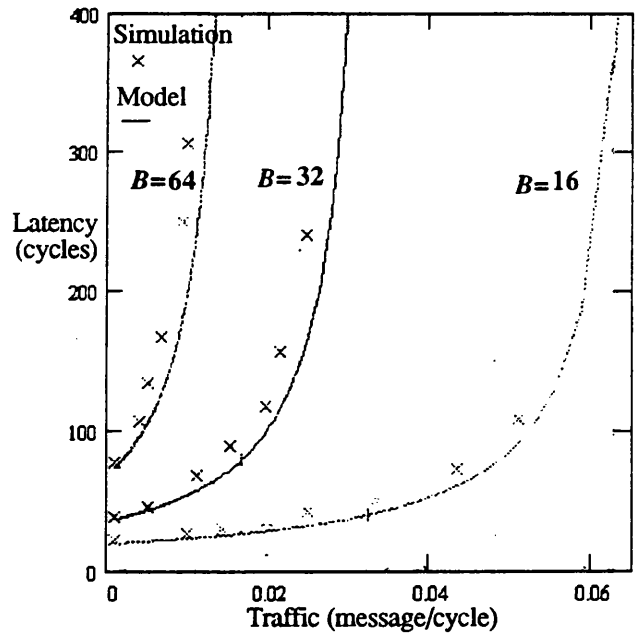
The discrete-event simulator, which verifies the above model, uses all the same assumptions as that of Section 3.2.1, except that assumption (d) is modified to

d') Routing is random.

Figures 3.12 and 3.13 shows network latency predicted by the queueing model compared with the simulator. Results are shown for various $N$ ($16^2$, $4^4$) and $B$ (16, 32, 64). As with restricted routing, the model underestimates latency under moderate and heavy traffic. However, the behaviour predicted by the model is very close to that of the simulator. (The network saturation point and the effects of dimensionality and width on network performance is the same as with restricted routing.)



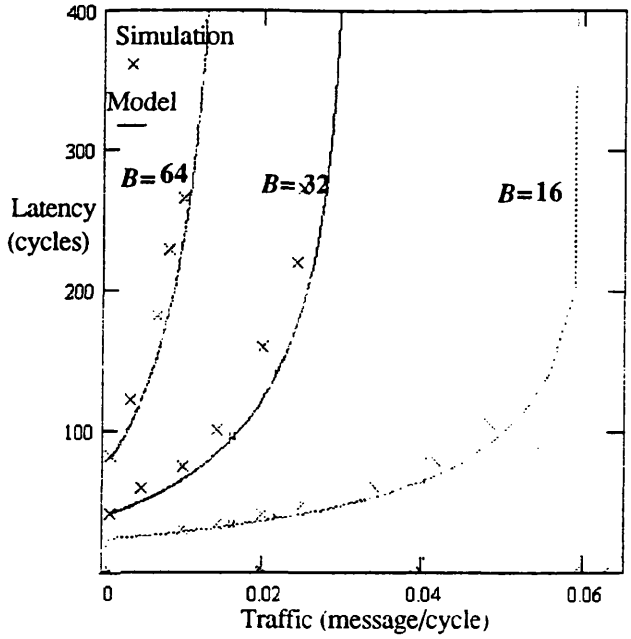

a) SAM



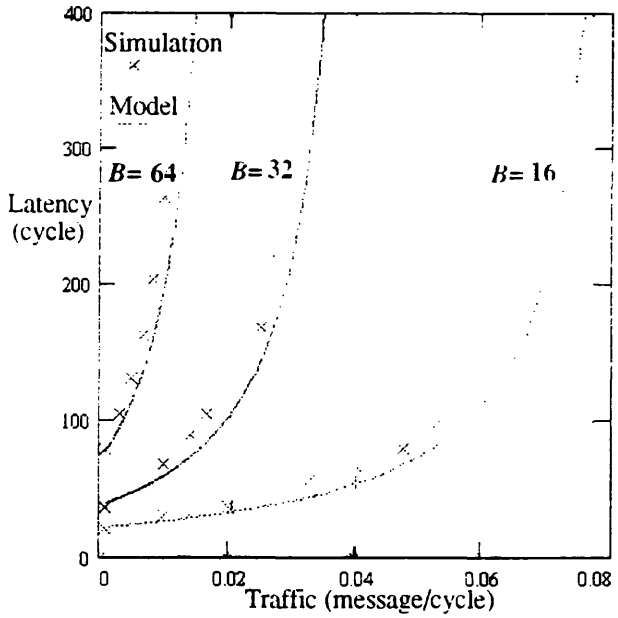

b) MAM

**Figure 3.12: Comparing the virtual cut-through model with simulation.**

***n*=2, *N*=256.**




a) SAM




b) MAM

**Figure 3.13: Comparing the virtual cut-through model with simulation. $n$=4, $N$=256.**

### 3.3.2 Restricted versus random routing

The results for restricted and random routing are shown for a 2, 4, and 12-D DCSH where $N$=4096 nodes and $B$=128 bits (the 12-cube is the highest possible dimension with $N$=4096. These dimensions were chosen to illustrate the impact of dimensionality on the performance of the routing algorithms). Figure 3.14 (a) and (b) show message delays with SAM, while Figure 3.15 (a) and (b) show those with MAM. The figures reveal that restricted routing has comparable or better performance than random routing depending on the dimensionality. To explain this, the traffic rates on network channels under random and restricted routing are first derived analytically. For the sake of the discussion, unit-length messages are assumed.

In random routing, the traffic rate on a network channel is given by

$$\rho^{Rand} = \frac{m}{n} d \tag{3.45}$$

The traffic rate on all network channels is identical.

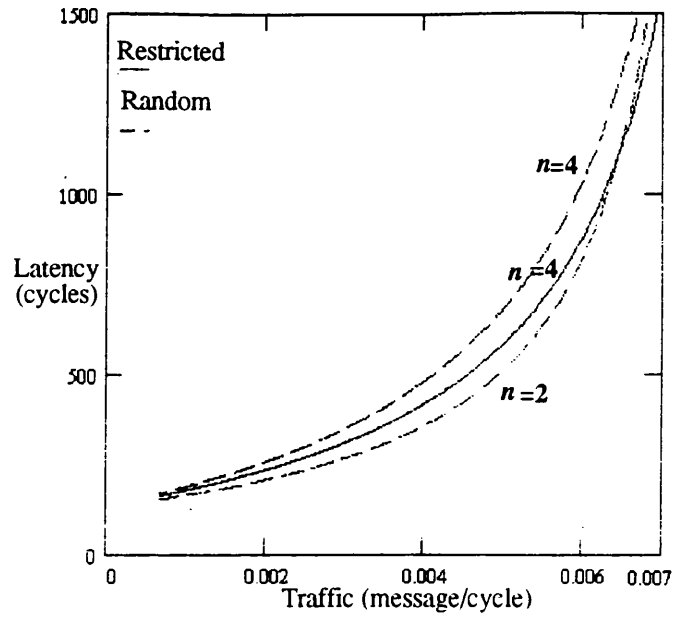
In restricted routing, messages visit dimensions in a predefined order. The traffic rate $\rho_i^{Rest}$ on channel $i$ ($1 \le i \le n$) can be determined as follows. In an $n$-D DCSH, a message routed across channel $i$ may have arrived from the local PE or from multiplexer $j$ ($i < j \le n$). To compute the rate that messages arrive at and hence leave from channel $i$, the rate that they are sent from the local PE and from all multiplexers must be determined.

For messages generated from the local PE, $(k\text{-}1)/k$ of these messages are directed through channel $n$. So,

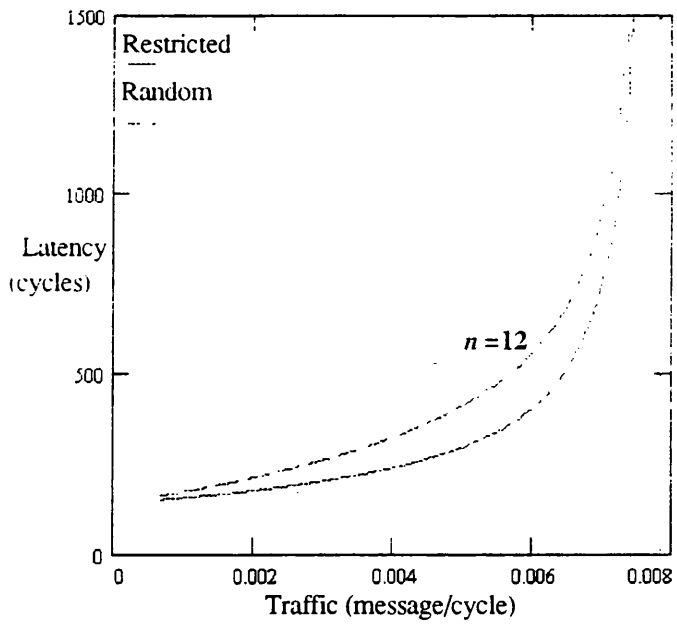
$$\text{P[message sent through channel } n \mid \text{originated from PE]} = \frac{k-1}{k} \tag{3.46}$$

In general, for $(1 \le i \le n-1)$

$$\text{P[message sent through channel } i \mid \text{originated from PE]} = \frac{(k-1)}{k^{n-i+1}} \tag{3.47}$$



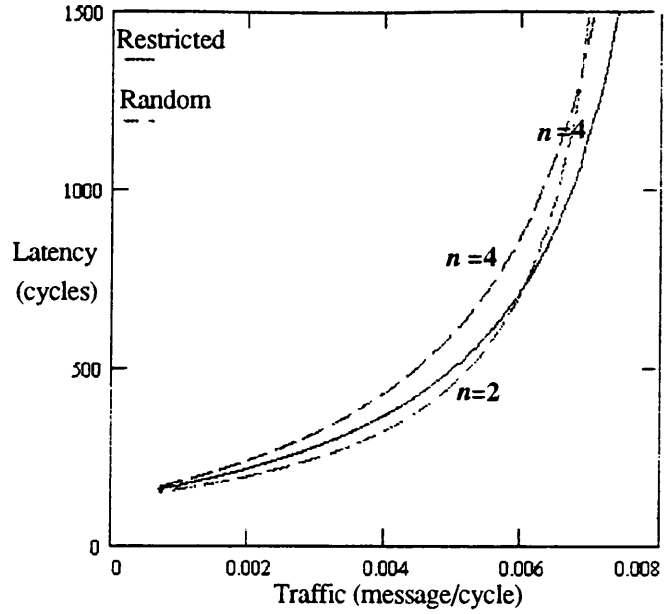

(a)



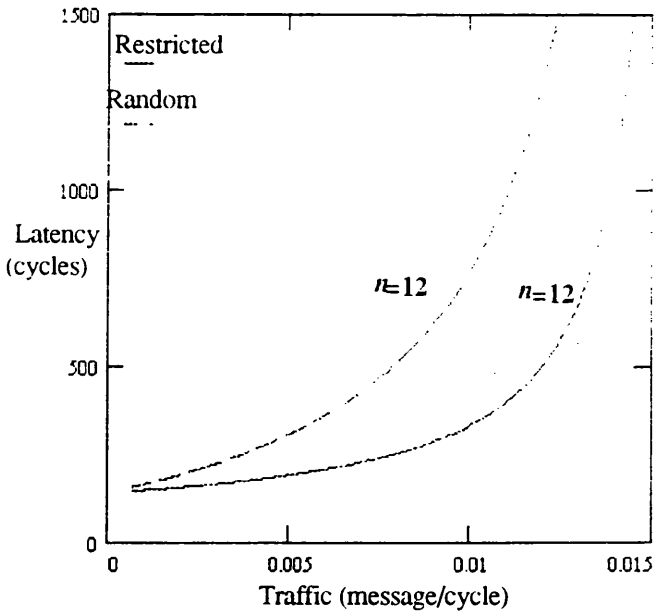

(b)

**Figure 3.14: Restricted versus random routing. $N$=4096, SAM[4].**

[4] There are only three traces in (a) because the results for restricted and random routing in 2-D DCSH are identical.




(a)




(b)

**Figure 3.15: Restricted versus random routing. $N$=4096, MAM$^4$.**

Similarly, the conditional probability that a message arriving at multiplexer $j$ will be sent through channel $i$ can be derived as

$$P[\text{message sent through channel } i \text{ from multiplexer } j] = \frac{(k-1)}{k^{j-i+1}} \tag{3.48}$$

To derive the probability that a message will be sent across channel $i$, note that channel $n$ is derived exclusively from the local PE. Given that the probability of the local PE generating a message in a cycle is $m$,

$$\rho_n^{Rest} = \frac{(k-1)}{k}\frac{(N-1)}{N}m \tag{3.49}$$

($N/(N-1)$ accounts for the fact that nodes do not reference themselves). A message generated from the local PE can be destined for any of the $(k-1)$ nodes at dimension $n$. Because of the uniform reference pattern, the probability that a given entry in multiplexer $n$ receives a message in a cycle is $\rho_n^{Rest}/(k-1)$. So the probability that multiplexer $n$ receives a message in a cycle is $\rho_n^{Rest}$.

The probability that in a cycle a message is sent through channel $n-1$, considering messages generated from both the local PE and multiplexer $n$, is

$$\begin{aligned}\rho_{n-1}^{Rest} &= \frac{(k-1)}{k^2}\frac{N}{(N-1)}m + \rho_n^{Rest}\frac{(k-1)}{k} \\ &= m\frac{(k-1)}{k}\frac{N}{(N-1)}\end{aligned} \tag{3.50}$$

Generally, the expression of $\rho_i^{Rest}$ $(1 \le i \le n-2)$ is given by

$$\rho_i^{Rest} = m\frac{(k-1)}{k^{n-i+1}}\frac{N}{(N-1)} + \sum_{i<j<n}\frac{(k-1)}{k^{i-j+1}}\rho_j^{Rest} \tag{3.51}$$

Solving the above equation reveals that for all $i$ $(0 \le i \le n-2)$

$$\rho_i^{Rest} = \frac{(k-1)}{k}\frac{N}{(N-1)}m \tag{3.52}$$

Equations 3.49, 3.50 and 3.52 show that in the traffic rate on all network channels is the same in restricted routing. Further, Equations 3.45, 3.49, 3.50 and 3.52 reveal that this traffic rate is equal to that in random routing.

In Section 3.2.1, to model the effects of restricted routing, it was assumed that messages crossing channel $i$ are either to be from the local PE or multiplexer $i+1$. Examining the above equations reveals that this is a reasonably realistic assumption to make since, in restricted routing, a message arriving at dimension $i+1$ switches to dimension $i$ with a higher probability compared with the switching probabilities to the other lower dimensions.

The reason why restricted routing is better than random is best understood by examining the message arrival rates at the different output queues. Without loss of generality, only the output queue corresponding to the lowest dimension is examined. In restricted routing, this output queue receives messages from ($n$-1) multiplexers. However, the contribution rates from the different multiplexers are disproportionate. Given that a message has arrived at the output queue, the probability is ($k$-1)/$k$ that the message arrived from the multiplexer of dimension 2. For $k$>>2, this probability is greater than 0.5. The multiplexer at dimension 2 alone contributes more than half of the total rate. All the remaining multiplexers in the higher dimensions contribute with lesser rates. On the other hand, with random routing, the contribution rates from the different multiplexers are identical. Given that a message has arrived at the output queue, the probability is $1/n$ that it arrived from a given multiplexer. It can be concluded that the probability of getting simultaneous message arrivals in restricted routing is lower, and this results in lower blocking at the output queue and ultimately lower message waiting times.

Figures 3.14 (a) and (b) reveal that with SAM when the dimensionality is low ($n \leq 4$), restricted routing has comparable latency to random routing under moderate traffic. When the dimensionality is high ($n$=12), restricted routing is better than random routing under moderate traffic. However, under high traffic the network saturation rate is the same for both routing strategies due to the bottleneck at the queue associated with the local channel. Figure 3.15 (a) and (b) reveal that with MAM for low-dimensional DCSHs ($n \leq 4$), the same behaviour is obtained as with SAM. However, for higher dimensional DCSHs ($n$=12), restricted routing has lower latency under moderate as well as high traffic.

In the 2-D DCSH, the latency with random and restricted routing is identical. Examining the different expressions for mean waiting times at the different queues shows that for $n$=2, the mean waiting time at a multiplexer and an output queue associated with either a network or local channel is the same under both routing schemes. The difference between the two routing strategies increases slightly as $n$ increases to 4 because the mean

waiting time at an output queue associated with a network channel is slightly higher under random routing due to the slight increase in the number of the competing input streams. However, when $n$ reaches 12, restricted routing has much better performance. This is due to the fact the mean waiting times at an output queue associated with a network channel is larger under random routing because of the higher number of competing input streams.

It can be concluded that low-dimensional DCSHs have the options of using either random or restricted routing since they provide comparable performance. High-dimensional structures, however, have to use restricted routing as it ensures a lower latency.

## 3.4 Alternative switching element designs

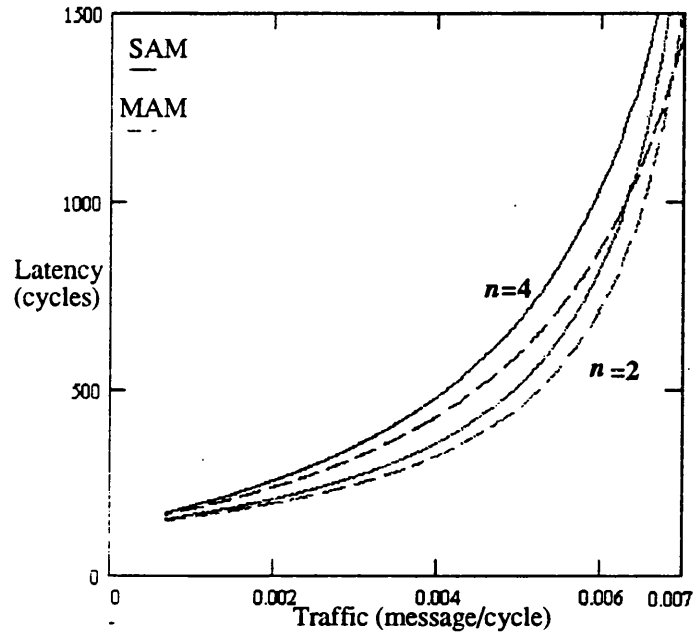
Section 3.2.1 presented two hardware models of the SE: MAM and SAM. Section 3.4.1 assesses the performance of the DCSH using these two designs. Further, in the SE design of Section 3.2.1, message queueing was provided at the input and output sides of the SE. Section 3.4.2 proposes a less-costly SE structure and evaluates its impact on performance.

### 3.4.1 Single versus multiple-accepting model

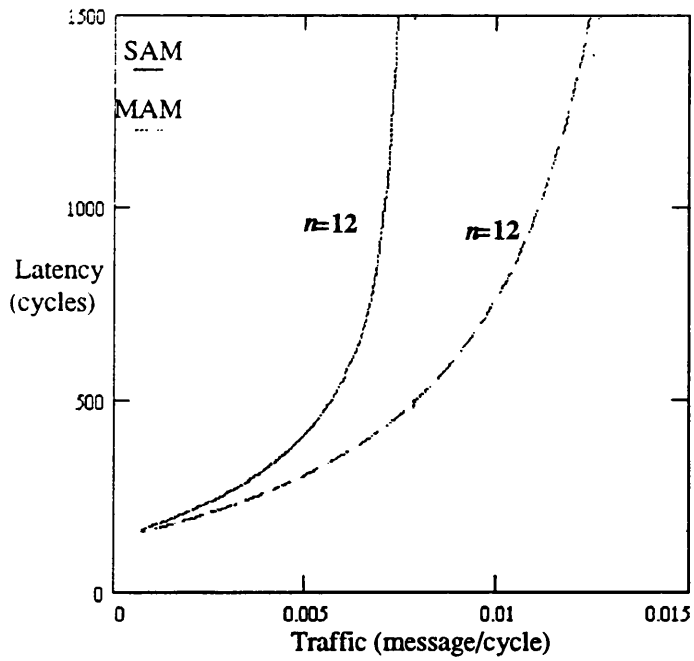
Using the queueing model developed in Section 3.2.1, latency results are shown in Figure 3.16 (a) for the 2 and 4-D DCSH and in Figure 3.16 (b) for the 12-D DCSH. The figures reveal that the relative merits of SAM and MAM depends on the dimensionality. As the dimensionality increases, the difference in performance between the two models gets larger. When the dimensionality is low ($n \leq 4$), the difference in performance is small. This is because the traffic rate on network channels is comparable to that to/from the PE. Hence, MAM does not improve performance significantly since the saturation rate of the queue leading to the PE is comparable to that of an output queue associated with a network channel. However, when the dimensionality is high ($n$=12), the difference is large as the traffic rate on network channels becomes lower than the traffic to/from the PE. The saturation rate of the queue leading to the PE is higher than that of the output queue associated with a network channel. MAM removes the bottleneck at the network-PE connection.

Low-dimensional DCSHs can use SAM without great loss in performance. Therefore, they can use SEs which are cheaper to implement. High-dimensional structures, and in

particular binary $n$-cubes, have to use MAM as it improves performance considerably. MAM, however, requires more costly SE design, thus making the cost of high-dimensional topologies more expensive as they already use larger number of channels than their lower-dimensional counterparts.




(a)




(b)

**Figure 3.16: Multiple versus single-accepting model. $N$=4096.**

### 3.4.2 Input queueing

Here, message buffers are provided only at the input side of the SE. The architecture is similar to that proposed for WR in Section 3.2.3, except that a input buffer can now hold entire messages if there is contention over the required channel.

For the sake of simplicity, the following analysis uses MAM and assumes that the traffic generated from each node follows an independent Poisson process with rate $m$ messages/cycle (messages are of equal length and uniformly distributed across the network). Furthermore, restricted routing is used. (The derivation of the queueing model with SAM is not presented due to its complexity, but the simulations indicate that the general conclusions do not change from those with MAM.)

Each node generates traffic with rate $m$ message/cycles into the network. As each node has $n$ output channels and a message travels on average $d$ hops, the traffic rate on a given channel is again

$$\rho = \frac{m}{n} dB \tag{3.53}$$

Simulation reveals that due to restricted routing the *effective* message service time (the time that a message, at the head of a queue, takes to leave the queue) does not increase and is equal to the message length. In other words, contention at a network channel is negligible. Therefore, to compute the mean message latency, only the mean message waiting time at the different input queues has to be considered. M/D/1 queueing theory gives the mean waiting time at the local PE input queue as [123]

$$w_l = \frac{mB^2}{2(1-mB)} \tag{3.54}$$

Recalling that the total message traffic at a multiplexed is $\rho$ messages/cycle, the mean waiting time at the multiplexer can be written as

$$w_m = \frac{\rho B^2}{2(1-\rho B)} \tag{3.55}$$

Again, this new model requires verification. The simulator uses the same assumptions as in Section 3.2.1. Figures 3.17 and 3.18 compare message delays given by the simulator and the above model for the 2, and 4-D topologies with $N$=$16^2$, $4^4$ and $B$=16, 32, 64

phits. They show that the above queueing model yields good predictions of message delay under light and moderate traffic. However, it overestimates latency under very high traffic because contention over the network channel is ignored. The model predicts latency well for the low-dimensional cases. However, the difference between the model and simulation gets larger as dimensionality increases. Nevertheless, ignoring such a contention factor still gives a good approximation which facilitates the building of a simple queueing model used for comparing different SE designs.

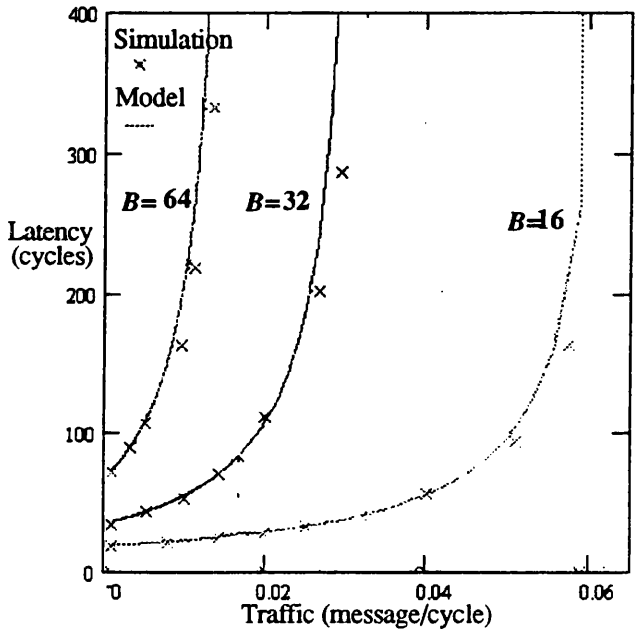



**Figure 3.17: Comparing the virtual cut-through model with simulations. $n$=2, $N$=256.**

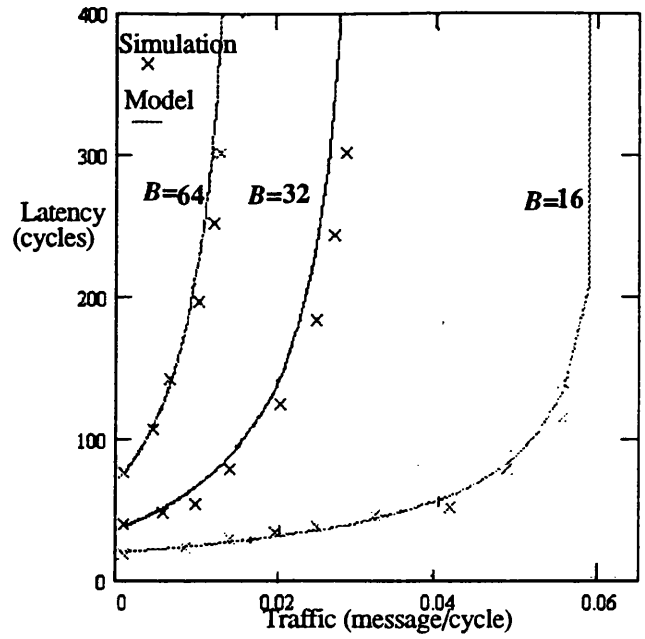



**Figure 3.18: Comparing the virtual cut-through model with simulations. $n$=4, $N$=256.**

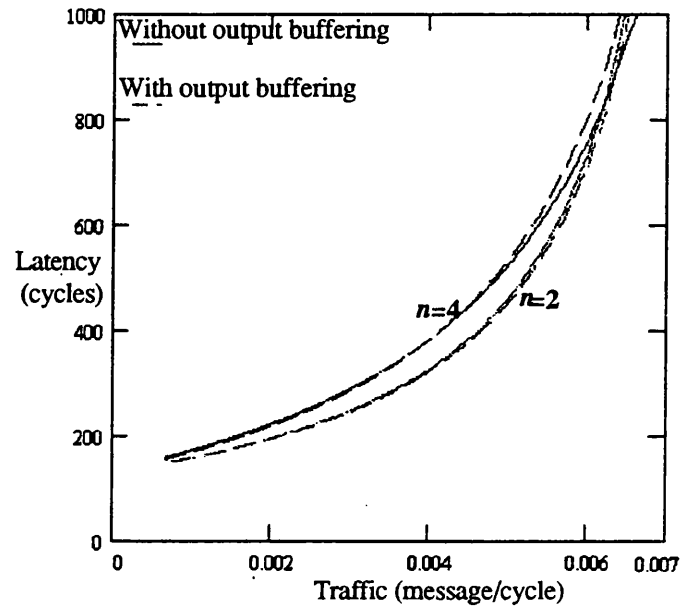
### 3.4.3 Merits of output queueing

Figures 3.19 (a) and (b) show the impact of output queueing on latency by comparing an SE which provides input and output queueing with one which provides only input queueing.

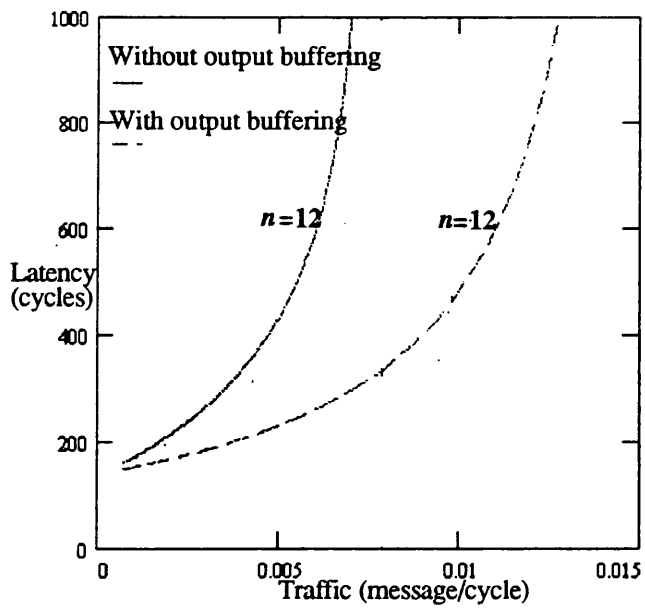
When $n$ is small ($n \leq 4$), the difference between the two designs is insignificant. This is due to the fact that in lower dimensional DCSHs, there is little channel contention since there is a small number of input streams competing over a network channel. Also, Section 3.3.2 has shown that with restricted routing the rates from the different input queues is uneven resulting in low channel contention.

When $n$ is high ($n$=12), the difference in performance between the two structures gets large. In the higher-dimensional topologies, there is a higher number of input streams, and thus an increase in channel contention. Where there is no output buffering a message may block another one even though it is going through a different channel. With output buffering, input and output sides are de-coupled from each other. If a message blocks another one, the blocking time is guaranteed to be no greater than the time required to copy a message into the required output queue.

Low-dimensional DCSHs can use SEs which provide only input buffering, and therefore allow for cheaper implementation and yet still deliver comparable performance to those which use more costly SEs which provide both input and output queueing. High-dimensional DCSHs, however, have to use SEs which provide both input and output queueing if acceptable performance is to be achieved.




(a)




(b)

**Figure 3.19: Impact of output buffering on performance. $N$=4096.**

## 3.5 Split-DCSHs

In an $n$-dimensional DCSH, a node has a single channel to communicate with the other ($k$-1) nodes in each cluster. However, at some additional implementation cost, the number of channels can be increased. Two possible DCSH variations were introduced in Chapter 2, namely the asymmetric-split-DCSH (AS-DCSH) and the symmetric-split-DCSH (SS-DCSH).

In the AS-DCSH, a channel is divided into two, each running in the opposite direction to the other; transmission in the different directions can then be concurrent. This retains the same wiring density as in the DCSH. In the SS-DCSH, a node has two channels per cluster. One channel is used (say) to communicate with odd-numbered nodes and the other with even-numbered ones. This configuration preserves network symmetry, but it has the drawback of requiring higher wiring density than the DCSH. The following sections assess the relative merits of the SS-DCSH and AS-DCSH. The models are presented for general message lengths and MAM (extensions of the model for SAM follows the same steps in Section 3.2.1).

### 3.5.1 Asymmetric split-DCSHs

An SE in the AS-DCSH has twice the number of internal switching paths and buffers as in the DCSH. Each network channel has its own output queue. If only one multiplexer were used per dimension, there would be little advantage in doubling the number of channels; the multiplexer would form a bottleneck because the traffic rate arriving at the multiplexer is higher than that at the network channels. Two multiplexers per dimension are therefore used, each of which can receive messages from the ($k$-1) cluster nodes. A message arriving along a dimension is copied into either multiplexer with equal probability.

As the AS-DCSH is asymmetric, the output queues along a cluster do not receive messages with equal rates except at nodes which are situated at the middle. Equation 3.2 gives the traffic rate ($\rho$) at an output queue in the DCSH. In the AS-DCSH, the traffic rate out of an SE $j$ ($1 \le j \le k$) along a cluster is also $\rho$. If the traffic rate at one output queue at SE $j$ is $\rho_j$, it is $(\rho - \rho_j)$ at the other output queue. Furthermore, along a cluster the traffic rate of one queue varies from 0 to $(\rho / 2)$ whereas the other one varies from $(\rho / 2)$ to $\rho$. The output queue which has the higher input traffic rate constitutes the bottleneck when the overall traffic is high. To simplify the analysis, effects due to the

network asymmetry are ignored. The traffic rate at all the queues, along a cluster, with the higher rate, is assumed to be the same, taken to be the average of $(\rho / 2)$ and $\rho$.

$$\rho' = \frac{\rho + \rho/2}{2} = \frac{3\rho}{4} \tag{3.56}$$

The waiting time at a multiplexer is the same as in the DCSH and is given by Equation 3.15. Using Equation 3.3 gives the waiting time at an output queue associated with a network channel as

$$w_o = \frac{\rho' B}{2(1-\rho')} \frac{((2n-1)P_t^2 - 2P_t + 1)}{2(n-1)} \tag{3.57}$$

The network latency predicted by the queueing model is validated against a simulator in Figures 3.20 and 3.21. The same assumptions as in Section 3.2.1 are used here again. Results are shown for various topologies ($16^2$, $4^4$) and message lengths (16, 32, 64). The figures reveals that ignoring the edge effect due to the network asymmetry in order to simplify the analysis still yields message latencies that are close to those produced by the simulator.

### 3.5.2 Symmetric split-DCSHs

Since an SE in the SS-DCSH has twice the number of channels as that of the DCSH, the traffic rate on a network channel is

$$\rho = \frac{md}{2n} B \tag{3.58}$$

The waiting times at a multiplexer and an output queue associated with a network channel are identical to Equations 3.15 and 3.57.

Figures 3.22 and 3.23 show network latency predicted by the model against simulations. The results reveal that the above model yields results that are close for all the tested cases.

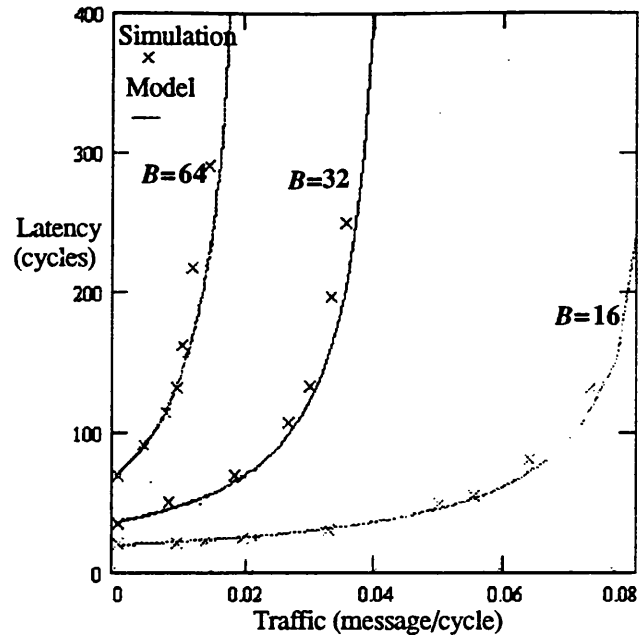



**Figure 3.20: Comparing the virtual cut-through model with simulations. MAM, $n$=2, $N$=256.**

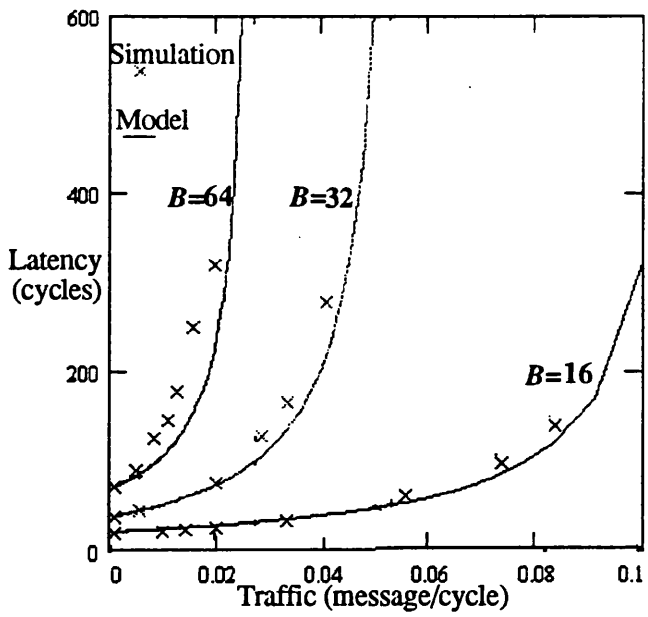



**Figure 3.21: Comparing the virtual cut-through model with simulations. MAM, $n$=4, $N$=256.**

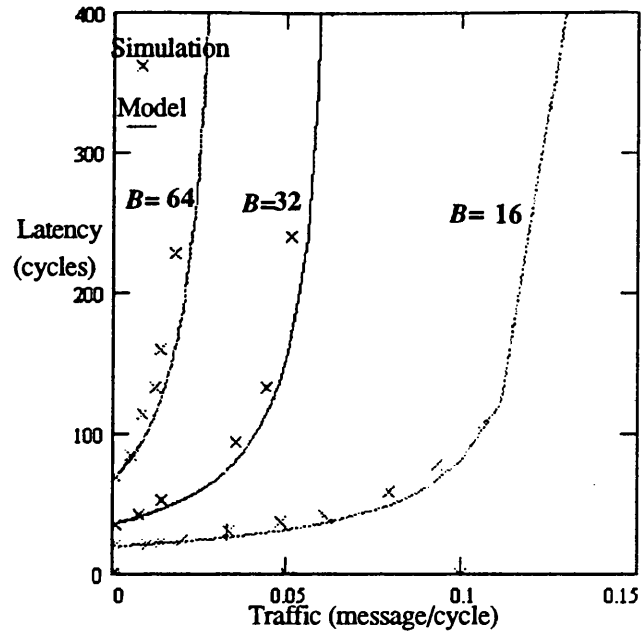



**Figure 3.22: Comparing the virtual cut-through model with simulations. MAM, $n$=2, $N$=256.**

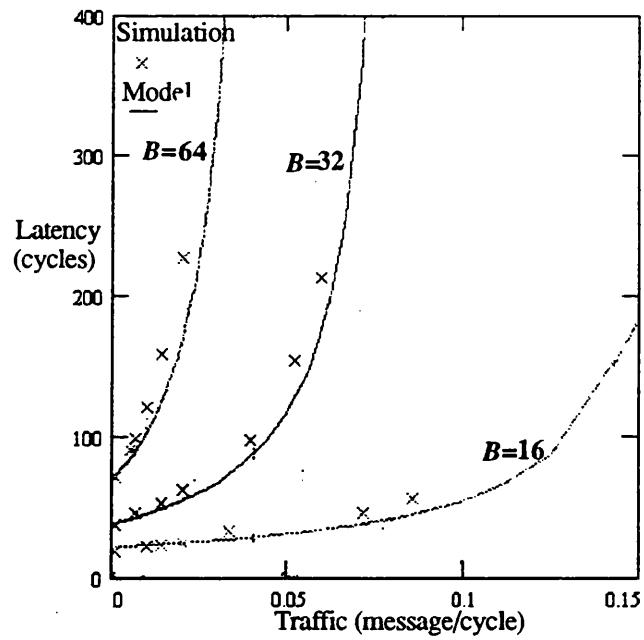



**Figure 3.23: Comparing the virtual cut-through model with simulations. MAM, $n$=4, $N$=256.**

### 3.5.3 Merits of split-DCSHs

Figure 3.24 compares the performance of the three DCSH arrangements, namely the DCSH, AS-DCSH, and SS-DCSH. The results are depicted for a 2-D case with $N$=4096 and $B$=128 (the conclusions are also applicable to the other higher-dimensional cases).

The SS-DCSH provides the best performance because it has the lowest traffic on network channels. The channel traffic rate in the SS-DCSH is 50% less than in the DCSH while in the AS-DCSH it is 25% less. At moderate traffic, the latency in the SS-DCSH is improved by 70% while in the AS-DCSH by 50%.

It is worth noting that with SAM, there is an improvement at moderate traffic. However, at high traffic all three structures saturate at the same traffic rate due to the bottleneck formed at the network-PE connection.

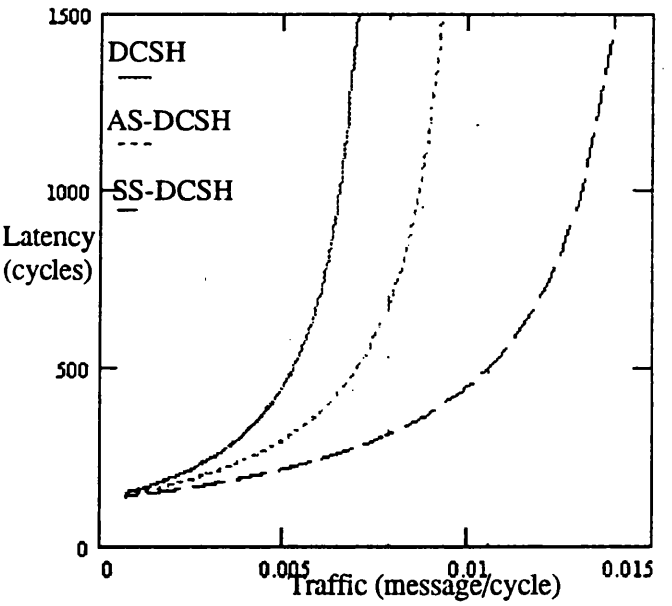



**Figure 3.24: DCSH versus split-DCSH. MAM, $n$=2, $N$=4096.**

## 3.6 Non-uniform traffic

So far, the performance of the DCSH has been examined under the uniform reference model. However, there exist parallel applications that exhibit non-uniform traffic pattern; a node references the other nodes with unequal probabilities. Typical forms of non-uniform traffic pattern include *communication locality* and *hot-spots*.

The DCSH can take advantage of communication locality inherent in several parallel applications [3, 54, 119, 120]. Generally, communication locality exists in a communication pattern when nodes that are close to the source are highly likely to be the destination nodes. There are several models for communication locality [3, 119]. The following sections focuses on the sphere of locality model [119]. In this locality model, an $n$-dimensional DCSH, of size $N=k^n$ nodes, is divided into disjoint groups of $N_c=k^c$ nodes each. All nodes within a group share the same sphere of locality, as composed of the nodes within that group. A message is destined for a node within the same sphere of locality as the source node with a probability $\beta$ and to a node in a different sphere with the probability $(1-\beta)$. The larger $\beta$ is, the stronger is the locality in communication. A message is destined for each node either within or to another sphere of locality with equal probability.

Pfister and Norton have described a hot-spot phenomenon where a portion of the network become saturated [127]. A hot-spot arises when a number of nodes direct a significant fraction of their generated traffic to a single destination; this often results in a substantial performance degradation. Global synchronisation, where each node in the system sends a synchronisation message to a distinguished node, is a typical situation which can produce hot-spots.

This section assesses the impact of non-uniform traffic on the performance of the DCSH. The following analyses use the same assumptions as Section 3.2.

### 3.6.1 Sphere of locality model

Let the size of the sphere of locality be $N_c = k^c$, $(n > c)$. The node address, which has a $n$ digit long radix $k$, is divided into two groups. The $c$ least significant digits identify nodes within the sphere whose $(n\text{-}c)$ most significant digits are the same. The other $(n\text{-}c)$ digits serve to identify a sphere. A sphere of locality is just a $c$-D DCSH within the $n$-D DCSH. A message generated from a node starts its journey from the $c$ lower dimensions

with the probability $\beta$, and from the ($n$-$c$) higher dimensions with the probability $(1-\beta)$.

## Virtual cut-through model

The same derivation steps as for the uniform case (Section 3.2.1) are used here. Let the channels be divided into two classes: sphere channels and non-sphere channels. Sphere channels are associated with the $c$ lower dimensions while non-sphere channels are associated with the ($n$-$c$) higher dimensions. The intra-sphere messages visit only sphere channels. Inter-sphere messages, on the other hand, visit sphere and non-sphere channels. The average number of sphere channels, which either an intra or inter messages visit, is given by

$$d_{sc} = c\frac{(k-1)}{k}\frac{N}{(N-1)} \tag{3.59}$$

Recalling that an inter-sphere message is not destined within the same sphere, the average number of non-sphere channels that an inter-sphere message visit is

$$d_{nsc} = (n-c)\frac{(k-1)}{k}\frac{N}{(N-N_c)} \tag{3.60}$$

Let $\rho_{sc}$ and $\rho_{nsc}$ be the probability that a message arrives on an incoming sphere channel and non-sphere channel respectively. Taking into account both intra and inter-sphere message visit $d_{sc}$ sphere-channels, $\rho_{sc}$ can be written as

$$\rho_{sc} = \beta\frac{mB}{c}d_{sc} + (1-\beta)\frac{mB}{c}d_{sc} = \frac{mB}{c}d_{sc} \tag{3.61}$$

Using the fact that inter-sphere messages visit, on average, $d_{nsc}$ non-sphere channels, $\rho_{nsc}$ is given by

$$\rho_{nsc} = (1-\beta)\frac{mB}{(n-c)}d_{nsc} \tag{3.62}$$

Since there are different traffic rates on sphere and non-sphere channels, the multiplexers and output queues at dimension $i$ $(1 \le i \le c)$, and those at dimension $j$ $(c < j \le n)$ have to be considered separately when computing the message latency. Equation 3.3 is used to compute the mean waiting time at the multiplexer, $w_{m_{sc}}$, and output queue, $w_{o_{sc}}$, at dimension $i$ $(1 \le i \le c)$ and they are found to be

$$w_{m_{sc}} = \frac{\rho_{sc}B}{2(1-\rho_{sc})}\frac{(k-2)}{(k-1)} \tag{3.63}$$

$$w_{o_{sc}} = \frac{P_{t_{sc}}(1-P_{t_{sc}})B\rho_{sc}}{(1-\rho_{sc})} \tag{3.64}$$

$$\text{where } P_{t_{sc}} = 1/d_{sc} \text{ if } (c>1) \text{ and } P_{t_{sc}} = 1 \text{ if } (c=1). \tag{3.65}$$

Similarly, the mean waiting time at the multiplexer, $w_{m_{nsc}}$, and output queue, $w_{o_{nsc}}$, at dimension $i$ $(c < i \leq n)$ can be written as

$$w_{m_{nsc}} = \frac{\rho_{nsc}B}{2(1-\rho_{nsc})}\frac{(k-2)}{(k-1)} \tag{3.66}$$

$$w_{o_{nsc}} = \frac{P_{t_{nsc}}(1-P_{t_{nsc}})B\rho_{nsc}}{(1-\rho_{nsc})} \tag{3.67}$$

$$\text{where } P_{t_{nsc}} = 1/d_{nsc} \text{ if } (n\text{-}1>c) \text{ and } P_{t_{nsc}} = 1 \text{ if } (n\text{-}1=c). \tag{3.68}$$

with SAM, the terminating message exits the network from the $c$ lower dimensions. Using Equation 3.3, the mean waiting time, $w_l$, at this output queue can be written as

$$w_l = \frac{P_{t_{cs}}(c-1)B\rho_{cs}}{2(1-cP_{t_{cs}}\rho_{cs})} \tag{3.69}$$

An intra-sphere message visits, on average, $d_{sc}$ sphere channels, and thus the mean intra-sphere latency is given by

$$L_{sc} = (1+w_{m_{sc}})d_{sc} + (1+w_{o_{sc}})d_{sc} + (1+w_l) + B \tag{3.70}$$

With MAM, the mean total intra-sphere latency is given by

$$L'_{sc} = (1+w_{m_{sc}})d_{sc} + (1+w_{o_{sc}})d_{sc} + B \tag{3.71}$$

An intra-sphere message visits on average $d_{nsc}$ non-sphere channels, the mean inter-sphere latency can be written as

$$L_{nsc} = (1+w_{m_{nsc}})d_{nsc} + (1+w_{o_{nsc}})d_{nsc} + B \tag{3.72}$$

With SAM, the mean total latency, after adding the mean intra-and inter-sphere latencies with their appropriate weights, is found to be

$$L = \beta L_{sc} + (1-\beta)(L_{sc} + L_{nsc}) \tag{3.73}$$

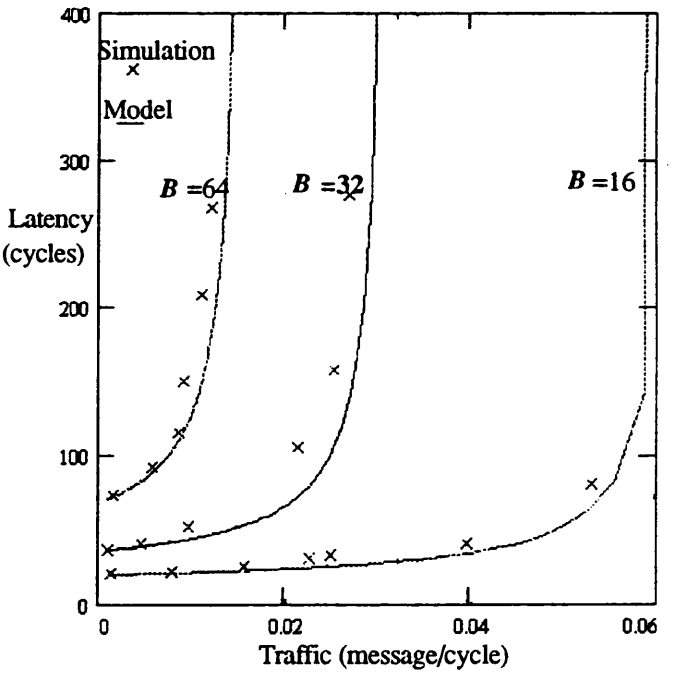
With MAM, the previous equation becomes

$$L' = \beta L'_{sc} + (1-\beta)(L'_{sc} + L_{nsc}) \tag{3.74}$$

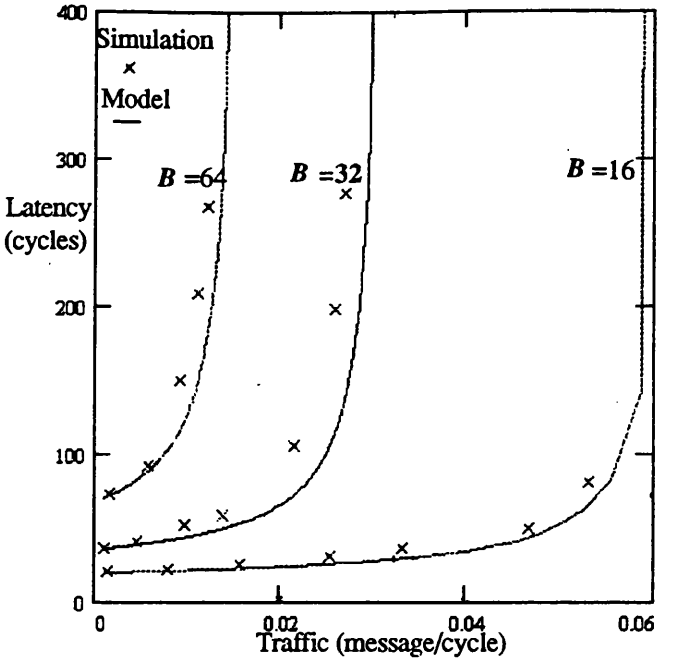
The above queueing model has been validated by mean of a simulator which uses almost the same assumptions as in Section 3.2.1 except that

c') A generated message is destined with the sphere of locality with probability $\beta$ and to another sphere message with probability $(1-\beta)$.

Figures 3.25 (a) and (b) show message delays of the model and those of the simulation for the 2-D DCSH with the following parameters: $N$=256 nodes, $B$=16, 32, and 64 phits, $c$=1 and $\beta$=50%. The queueing model predicts message latency reasonably accurately under light and moderate traffic. However, it overestimates latency under high traffic because of the synchronised-SE assumption.




a) SAM




b) MAM

**Figure 3.25: Comparing the virtual cut-through model with simulation. $n$=2, $N$=256, $c$=1, β=50%.**

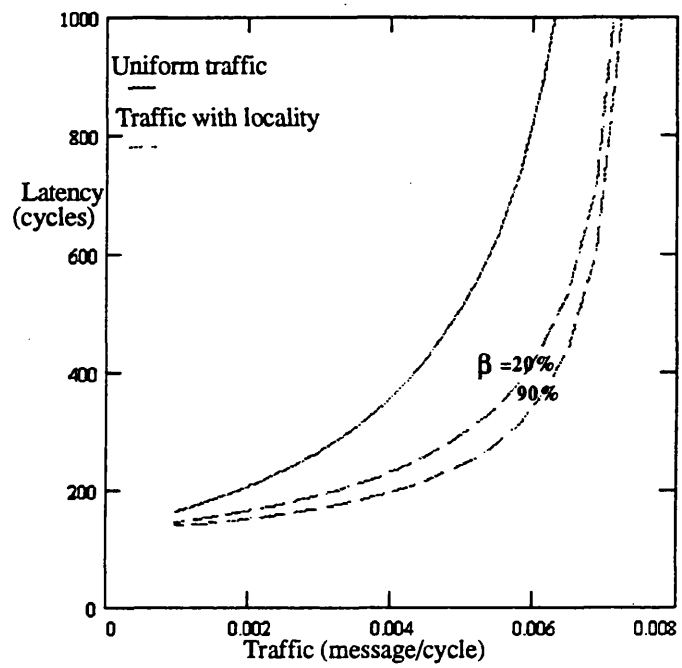
## Virtual cut-through performance

For the sake of clarity, only the 2-D case is examined. Results are presented for $N$=4096 and $B$=128 phits. The size of the sphere of locality $c$=1 with β=20% and 90%. The uniform traffic model is also shown in order to assess the gain of exploiting locality.

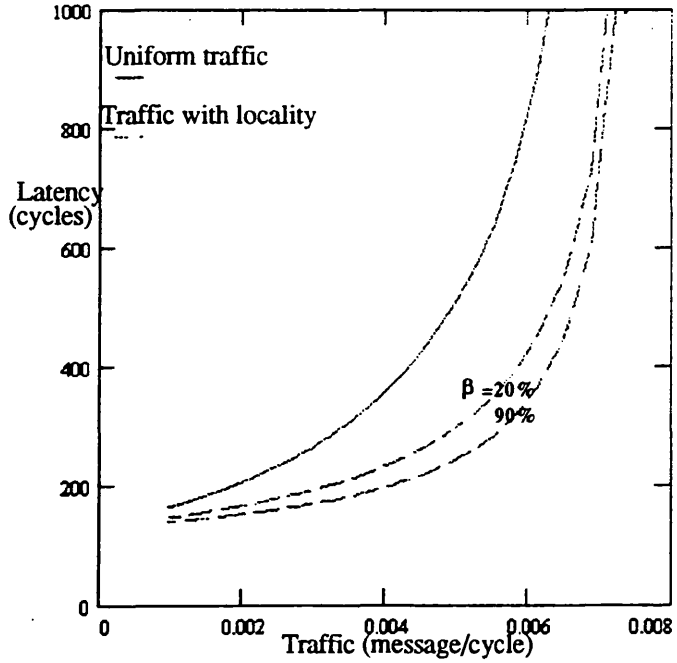
Figure 3.26 (a) which shows message delays with SAM, reveals that as communication locality increases, the latency is reduced compared to the uniform model under light and moderate traffic. This is because with locality, the traffic rate on sphere channels and non-sphere channels is lower than that with the uniform traffic. Messages at the output queue leading to the PE do not experience any delay since there is only one stream arriving at the queue. The network saturates at comparable traffic rates with both uniform traffic and communication locality. With the former pattern, the network saturates at $m_s$=1.01. With communication locality, on the other hand, sphere channels handle both inter and inter-sphere messages and therefore saturate earlier than non-sphere channels. The sphere channels saturate at $m_s$=1.

Figure 3.26 (b) reveals that with MAM the results are similar to those with SAM. For higher-dimensional DCSHs, if the size of the sphere of locality is greater than unity,

SAM improves the performance under moderate traffic whereas MAM improves it under moderate and high traffic.




a) SAM




b) MAM

**Figure 3.26: Impact of the sphere of locality on performance. $n$=2, $N$=4096, $B$=128, $c$=1.**

## Wormhole routing model

The same assumptions and procedure of the uniform case (Section 3.2.3) are used again. When the traffic is light, after adding the latency of intra and inter-sphere messages with their appropriate weights, the mean total latency is found to be

$$\begin{aligned} L &= \beta d_{sc} + (1-\beta) d_{nsc} + (1-\beta) d_{sc} + 1 + B \\ &= d_{sc} + (1-\beta) d_{nsc} + 1 + B \end{aligned} \tag{3.75}$$

where $d_{sc}$ and $d_{nsc}$ are given by Equations 3.59 and 3.60. As traffic increases, dimension $i$ $(1 \le i \le c)$ and $(c < i \le n)$ have to be considered separately as traffic rates in these two sets of dimensions are different. Since both inter and intra-sphere messages visit dimension $i$ $(1 \le i \le c)$, the traffic rate at this dimension is

$$m_{sc} = m \frac{N}{(N-1)} \tag{3.76}$$

As only inter-sphere messages visit dimension $i$ $(c < i \le n)$, the traffic rate at this dimension is given by

$$m_{nsc} = (1-\beta) m \frac{N}{N - N_c} \tag{3.77}$$

For dimension $i$ $(0 \le i \le c)$, Equation 3.23, 3.35, and 3.36 (Section 3.2.3) become

$$L_{sc_0} = B \tag{3.78}$$

$$L_{sc_i}^M = \frac{(1-\alpha) m_{sc} L_{sc_i}^2}{2} \tag{3.79}$$

$$\begin{aligned} L_{sc_{i+1}} = L_{sc_i} &+ (1-\alpha) L_{sc_i}^M + \alpha (1-\alpha)^3 m_{sc} (L_{sc_i} + L_{sc_i}^M)^2 \\ &+ \alpha^3 (1-\alpha) m_{sc} L_{sc_i}^2 \end{aligned} \tag{3.80}$$

Similarly, for dimension $i$ $(c < i \le n)$, they become

$$L_{nsc_i}^M = \frac{(1-\alpha) m_{nsc} L_{nsc_i}^2}{2} \tag{3.81}$$

$$\begin{aligned} L_{nsc_{i+1}} = L_{nsc_i} &+ (1-\alpha) L_{nsc_i}^M + \alpha (1-\alpha)^3 m_{nsc} (L_{nsc_i} + L_{nsc_i}^M)^2 \\ &+ \alpha^3 (1-\alpha) m_{nsc} L_{nsc_i}^2 \end{aligned} \tag{3.82}$$

Message delays of the queueing model and those of the simulator for the 2-D DCSH are shown in Figure 3.28. The same assumptions as in Section 3.2.3 are used except that

c') A generated message is destined with the sphere of locality with probability β and to another sphere message with probability $(1-\beta)$.

The parameters are set as with VCT. The queueing model predicts message latency up to moderate traffic. However, the disparity between the two models get larger as traffic increases. Nevertheless, the above model can be used as a tool to predict and evaluate the impact of locality on the performance of the DCSH.

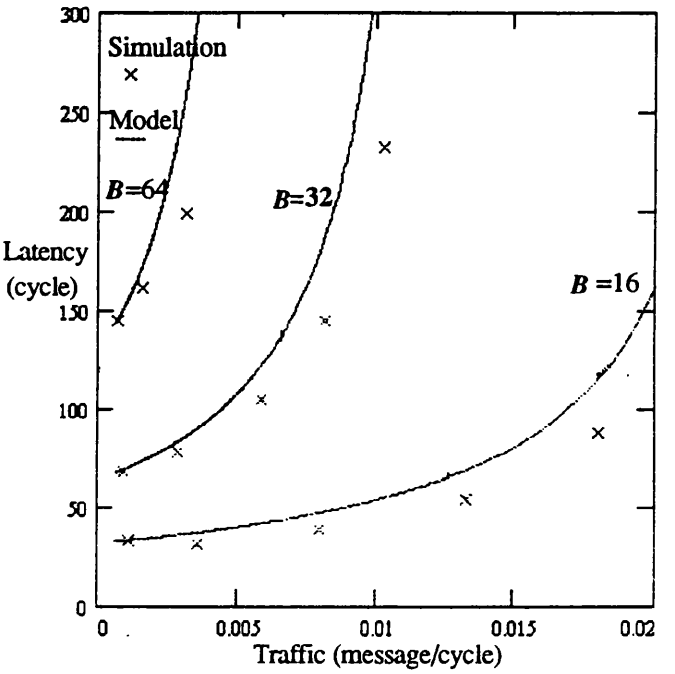



**Figure 3.28: Comparing the wormhole routing model with simulation. $n$=2, $N$=256, $c$=1, β=50%.**

## Wormhole routing performance

Figure 3.29, which shows performance results where the same parameters as with VCT are used, reveals that exploiting locality improves performance considerably. As β increases, the percentage of local traffic increases. More intra-messages go through fewer hops, and thus go through fewer blocking stages. Further, inter-sphere messages experience less blocking as the traffic at dimension $i$ $(c < i \leq n)$ decreases. As a result, the mean message latency is reduced.

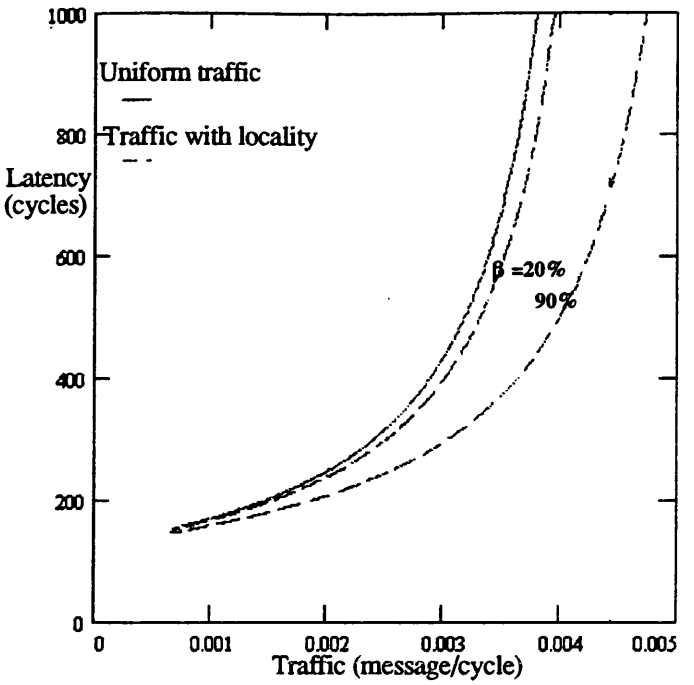



**Figure 3.28: Impact of the sphere of locality on performance. $n$=2, $N$=4096, $B$=128, $c$=1.**

## 3.6.2 Hot spots

Hot spot effects have been examined in the DCSH by directing a fraction $h$ of generated from each node to a particular node in the system (referred to as hot-traffic), while the remaining fraction (1-$h$) is distributed uniformly across all the nodes (referred to as background traffic). This section reports briefly on the results obtained through simulations of a 256 node DCSH.

For VCT, the results are identical to those reported by Abraham for the binary $n$-cube [28, 34]. Hot-spot traffic suffers considerable performance degradation with restricted routing. With random routing, however, the hot-spot traffic suffers substantial waiting delays. This cannot be avoided given the nature of hot spot traffic. However what is interesting is the small effects of hot-spot traffic on the background traffic.

For WR, the results are similar to those reported by Ade and Vernon for meshes and tori [128]. Only experiments with restricted routing have been conducted because random routing requires the use of virtual channels to avoid message deadlock and therefore requires a different SE design from that described in Section 3.2.3. Hot-spot traffic degrades system performance considerably because the queues at the source node become the bottleneck. They build up to infinity as the hot-spot traffic blocks the

background traffic. If random routing is used together with virtual channels to avoid message deadlock, and different queues at the source are provided, each for a given network channel, the effect of hot-spot on system performance is reduced and therefore does not impact significantly on the background traffic [128].

## 3.7 Conclusions

Different switching methods have been analysed for the DCSH. Virtual cut-through provides the optimum performance. Wormhole routing is more cost-effective than virtual cut-through under light traffic since it has the same performance as virtual cut-through and has the advantage of using cheaper switching elements. Message switching suffers from lower performance under light traffic compared to virtual cut-through and wormhole routing. However, it employs simpler switching elements than virtual cut-through and wormhole routing. Further, message switching can be more cost effective in lower-dimensional DCSHs when messages are short, as it introduces tolerable performance degradation.

With virtual cut-through, restricted routing provides much better latency than random routing for higher-dimensional DCSHs such as binary $n$-cubes, while the difference in performance is insignificant for the lower-dimensional ones. Therefore, the latter can use random routing to reduce their vulnerability to channel and node failures.

Different hardware designs for the switching element have been considered. When the single-accepting model is used, the performance bottleneck resides at the queue situated at the switching element-processing element interface. The multiple-accepting model removes this bottleneck and improves performance considerably when the dimensionality is high. However, little gain is obtained when the dimensionality is low. This finding reveals that low-dimensional DCSHs can reduce their implementation cost by using the single-accepting model, while they can still handle traffic capacity which exactly matches that offered by the processing elements.

The provision of output buffering at a switching element does not yield any performance gain when the dimensionality is low. On the other hand, providing such buffering reduces latency considerably when the dimensionality is high, as in the binary $n$-cube. This shows that low-dimensional DCSHs have the important advantage of being able to use cheap switching elements without any degradation in performance.

Different DCSH variations, notably the SS-DCSH and AS-DCSH, have been evaluated. The former, which maintains network symmetry, yields the best performance. However, this is achieved at the expense of higher wiring density. Although the latter, which does not preserve symmetry, has a latency that lies between that of the DCSH and SS-DCSH; it offers a decisive advantage in that it requires only the same wiring density as the DCSH.

Finally, this chapter has also examined the impact of non-uniform traffic on the DCSH performance, namely communication locality and hot-spots. With virtual cut-through, locality yields performance gain under moderate traffic when the single-accepting model is used. However, to get maximum gain under moderate as well as high traffic, the multiple-accepting model must be used, particularly in higher-dimensional DCSHs. With wormhole routing, on the other hand, communication locality improves performance since it reduces message blocking. Hot-spots under restricted routing cause performance degradation with both virtual cut-through and wormhole routing. However, their effects are considerably reduced by using random routing.

# 4 Alternative hypermesh implementations

## 4.1 Introduction

Having developed a model of DCSH networks, this chapter compares these to other hypermesh implementations, notably the spanning-bus hypercube, crossbar switch hypermesh, and generalised hypercube, taking into account the bandwidth constraints imposed by VLSI and multiple chip technology.

The outline of the chapter is as follows. Sections 4.2, 4.3, and 4.4 deal with the spanning-bus hypercube, crossbar switch hypermesh, and generalised hypercube respectively. In each case, queueing models for virtual-cut through (VCT) and wormhole routing (WR) are developed. A performance comparison with the DCSH is presented assuming equal implementation cost. Throughout this chapter, the effect of decision time on network performance is ignored as it has the same effects in all these hypermesh implementations.

## 4.2 Spanning-bus hypercubes (SBHs)

A bus in the SBH is time-multiplexed among the cluster nodes. Before transmitting a message, a node has to gain access to the shared bus. Once it is granted access, it uses

the full bus bandwidth to perform message transfer. The multi-access spanning bus hypercube is an example which uses this implementation scheme [91, 111].

There are two delay factors associated with the shared-bus implementation, namely the *bus-arbitration* and *bus-release* times [108, 129]. The first of these is due to the fact that a bus has to choose which requesting node is granted access. The second arises because a new bus master has to wait until there is no signal propagating anywhere on the bus before it can initiate a transmission. Real bus systems can support only a relatively small number of nodes because performance degrades due to the increased bus contention as the network size increases. Further, significant performance overheads are incurred when there are frequent changes of bus mastership [108].

### 4.2.1 Mathematical models for SBHs

The following analyses assume that restricted message routing in the $k^n$ node SBH is used because this routing strategy makes a high performance SE simple to implement [4, 65, 76]. A node generates a message, with probability $m$ in a cycle, of a fixed length ($B$ flits) and destined to the ($N$-1) nodes with equal probability.

#### Virtual cut-through model

Figure 4.1 shows the basic SE structure for VCT. Instead of a ($k$-1)-to-1 multiplexer at the input side of the SE as in the DCSH (Section 3.4.1), there is an input controller which receives messages and directs them to the appropriate output queue.

The same assumptions as Section 3.2.1 are used here. The model is derived for the single-accepting model (SAM), but its adaptation for the multiple-accepting model (MAM) is also shown. The model is briefly described for general sized messages. Extensions to include the effects bus-arbitration and release times are then presented.

The fact that an SE is connected to $n$ buses and messages visit $d$ buses on average ($d$ is the same as in the DCSH), gives the traffic rate on bus $i$ ($1 \le i \le n$)

$$\rho = \frac{m}{n} dB \tag{4.1}$$

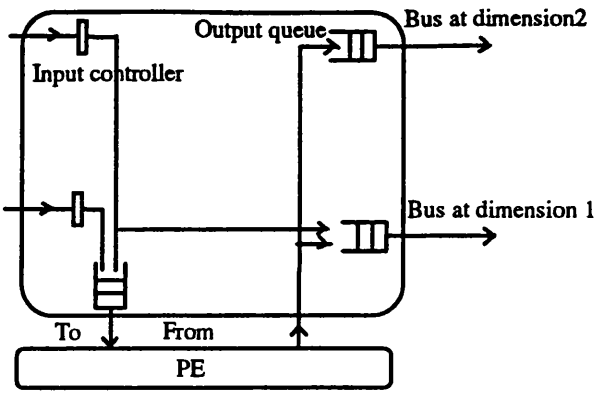



**Figure 4.1: The SE structure in the 2-D SBH using virtual cut-through.**

To make the analysis more tractable, the $k$ output queues that compete for bus $i$ are treated as a single queue with $2k$ input streams; $k$ streams have a rate of $\rho_t$ phits/cycle coming from the local PE as given by Equation 3.10, and another $k$ streams with rate of $\rho_s$ coming from the previous dimension, as given by Equation 3.11. Equation 3.3 gives the mean waiting time at an output queue, including the delay due to bus contention, as

$$w_o = \frac{(k-1+2P_t-2P_t^2)\rho B}{2(1-k\rho)} \tag{4.2}$$

$P_t = 1/d$ is the probability of termination. With SAM, the mean waiting time at the output queue associated with the local channel is given by Equation 3.17. After adding the time required to transmit a $B$ phit message, the mean total latency can be written as

$$L = (1+w_o')d + (1+w_l') + B \tag{4.3}$$

(With MAM, the second term in the left hand side of the above equation is removed).

To include the bus-arbitration and bus-release times, which can be lumped together as one factor, $D_b$, the message service time has to increase by $D_b$ cycles when crossing a bus. Hence, Equations 4.1, 4.2 and 4.3 become

$$\rho' = \frac{m}{n} d(B+D_b) \tag{4.4}$$

$$w_b' = \frac{(k-1+2P_t+2P_t^2)\rho'}{2(1-k\rho')}(B+D_b) \tag{4.5}$$

$$L' = (1+w_o')d + (1+w_l') + B + D_b d \tag{4.6}$$

## Wormhole routing model

Figure 4.2 shows the SE structure for WR. As with the DCSH, input buffering of a few flits capacity (e.g., 2 flits) is provided only at the input side of the switching element. The same assumptions and procedure of Section 3.2.3 are used here again.

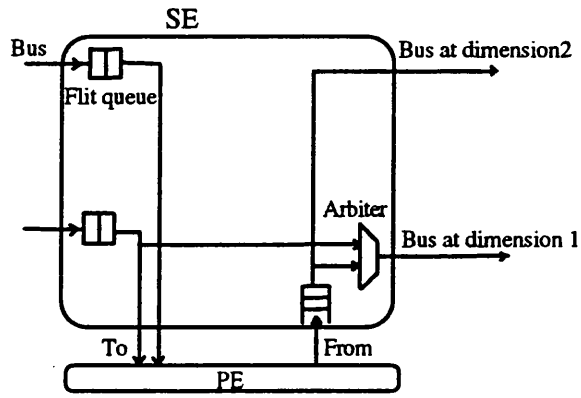



**Figure 4.2: The SE structure in the 2-D SBH using wormhole routing.**

As shown in Figure 4.3, the different traffic rates that enter or skip a dimension is the same as in the DCSH (Equations 3.24-3.30). The traffic that passes through a dimension has to compete for the corresponding bus and therefore may encounter blocking delay. Figure 4.4 shows the bus contention model in a dimension. There are $m_p = (1-\alpha)m'$ messages passing through dimension $i$, and therefore trying to access bus $i$, from each of the $k$ nodes. Thus a given message sees, on average,

$$m_b = (k-1)(1-\alpha)m' \tag{4.7}$$

messages from the other ($k$-1) nodes competing for the same bus. Equation 3.31 gives the latency seen by a message when accessing bus $i$ as

$$L_i^B = \frac{m_b L_i^2}{2} \tag{4.8}$$

With the model depicted by Figure 4.3, the latency seen by a message entering dimension $i$, including the bus contention as well as the latency due to routing at the subsequent dimensions with their appropriate weight, can be written as

$$L_{i+1} = L_i + (1-\alpha)L_i^B + \alpha^3(1-\alpha)m'L_i^2 + \alpha(1-\alpha)^3 m'(L_i + L_i^B)^2 \tag{4.9}$$

The bus-arbitration and release times ($D_b$) are included by simply increasing the message service time when crossing a bus by a factor $D_b$. Thus, Equation 4.9 becomes

$$L_{i+1}^{B'} = D_b + \frac{m_b(L_i + D_b)^2}{2} \tag{4.10}$$

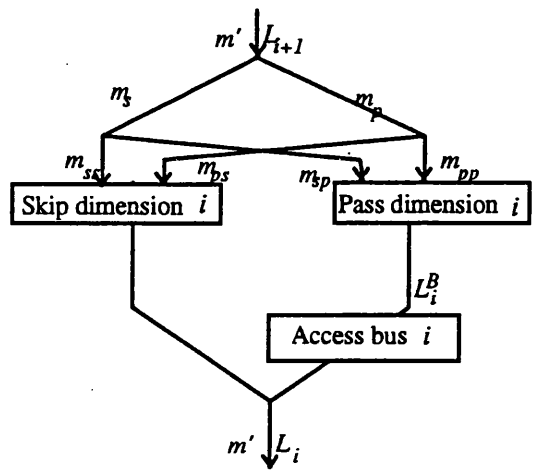



**Figure 4.3: A model to determine the latency in a SBH dimension.**

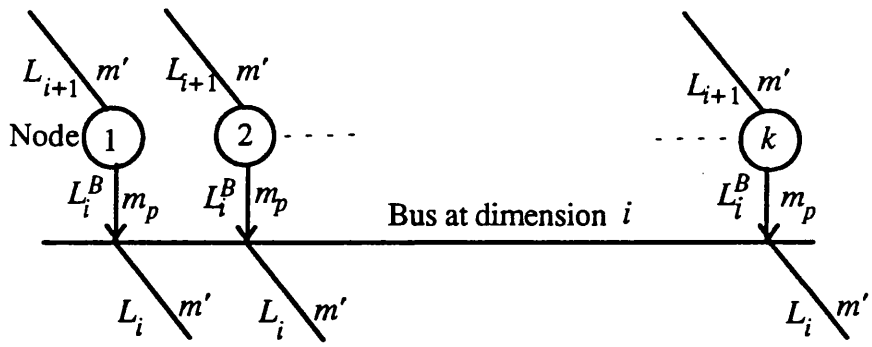



**Figure 4.4: The bus contention model in a dimension.**

### 4.2.2 Comparison of SBHs and DCSHs

As discussed in Chapter 1, implementation cost dictates the width of a network channel [22, 23, 34, 37]; a topology with a rich interconnect pattern is likely to have narrower channels than that with a simple interconnect pattern. Two cost measures have been identified in the literature each suitable for a particular technology; notably wiring density and pin-out for VLSI and multiple-chip technology respectively [22, 23, 34, 37].

The following sections compare the performance of an SBH to its DCSH counterpart. The discussion focuses on a low-dimensional case, $n$=2 (higher-dimensions are considered if they yield different conclusions). It is assumed that the channel width in the DCSH is one phit. Thus the message lengths which are considered below represent the message aspect ratios for the DCSH. The channel width in its SBH counterpart is constrained in such a way to maintain equal implementation cost with the DCSH. Performance results are presented for $N$=4096, which represents a moderately large system. The discussion is focused on MAM (results for SAM are discussed if they are different).

#### Wiring density

Systems implemented on VLSI chips are wire limited; as the number of components to be interconnected becomes large, the area occupied by the interconnecting wires themselves becomes the major limiting factor [1, 2, 22, 37]. Bisection width has often been used to measure the network wire density [22, 36, 37]; other measures have also been proposed such as the *peak width* [98, 130], the maximum number of channel crossings along one physical dimension. The latter may be higher than the bisection width in networks, such as binary $n$-cubes, and is strictly more accurate than the bisection width. However, in general the bisection width acts as an acceptable approximation. To compute the bisection width of a network, the following assumptions are made [22, 37].

a) The network is embedded into the 2-D plane such that $\sqrt{N}(=k^{n/2})$ nodes are in a given row or a column.

b) Nodes are placed in such a way that $n/2$ logical dimensions are embedded in each of the two physical dimensions.

In a $k^n$ node SBH, the buses crossing the midpoint of the network are all in the highest dimension [22, 37]. Since there is one bus per dimension and in a 2-D layout there are

$k^{n/2}$ rows, the bisection width of the SBH is given by

$$W_{SBH} = k^{n/2} k^{n/2-1} W_{SBH} = \frac{N}{k} W_{SBH} \tag{4.11}$$

The channels crossing the midpoint of the network in its DCSH counterpart are all in the highest dimensions. In a row, each of the $k^{n/2}$ nodes has a channel crossing the midpoint. Therefore, considering all the $k^{n/2}$ rows together, the bisection width of the DCSH can be written as

$$W_{DCSH} = k^{n/2} k^{n/2} W_{DCSH} = N W_{DCSH} \tag{4.12}$$

An SBH has the same wiring density as a DCSH if its channel width satisfies

$$W_{SBH} = k W_{DCSH} \tag{4.13}$$

The bus in an SBH is therefore $k$ times wider than a channel in a DCSH.

### Virtual cut-through performance

Figure 4.5 depicts the mean latency as a function of the generation rate in the SBH and DCSH, where the message length was selected at $M$=128 phits, which is a typical message length in fine-grain computation, as has been suggested by several researchers including Dally, Abraham, Agarwal, and Scot *et al* [22, 37, 52, 51]. Even when the bus-arbitration and release times are only one cycle, the DCSH has better performance than its SBH counterpart. Figure 4.5 reveals the sensitivity of the SBH to bus contention; a slight increase in message service time results in considerable performance degradation. With the unrealistic assumption of zero bus-arbitration and release times, the SBH outperforms the DCSH under light and moderate traffic. The wider bus of the SBH makes its message aspect ratio lower than that of the DCSH, and therefore reduces the low-load latency. However, and more importantly, Equations 3.4 and 4.14 reveal that both networks still saturate at the same traffic rate. This is because the benefits of a wide bus in the SBH is offset by high bus contention under high traffic.

The results in Figure 4.6 show the effect of message length on the system performance. Increasing the message length increases the delay by increasing the waiting times at the different queues. Further, changing the message length also affects the system load though the generation rate has been fixed. When the bus-arbitration and release times are two cycles, the DCSH has better latency characteristics than the SBH at all message lengths. Even when the bus arbitration and release times are ignored, the DCSH manages

to outperform the SBH when the message length is smaller than the bus width (see Figure 4.7). This is because the effective channel width ratio between the two networks is equal to the message length and in this case is less than the real ratio. As a result, the SBH cannot use its bus bandwidth efficiently; bandwidth is wasted. When the message length is greater than the bus width, the SBH has a lower latency because the effective channel width ratio is large enough to enable it to outperform the DCSH.

There are two situations where the message aspect ratio in the SBH can be smaller than the bus width. Firstly, when wiring density or pin-out counts become high, the bus in the SBH becomes much wider than a channel in the DCSH, and consequently becomes larger than the message aspect ratio. Secondly, as the network size increases and the message length is kept constant, the message aspect ratio in the SBH can become smaller than the bus width since this becomes wider than a channel in the DCSH under constant wiring density and pin-out constraints. The DCSH can take more advantage of these situations than the SBH as it can use its channel width more efficiently.

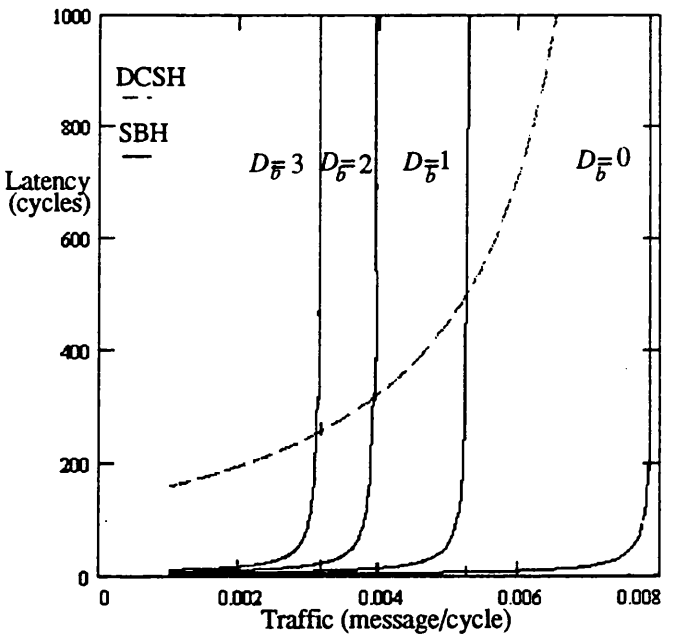



**Figure 4.5: Virtual-cut through performance in the SBH & DCSH for a fixed message length ($M$=128 phits).**

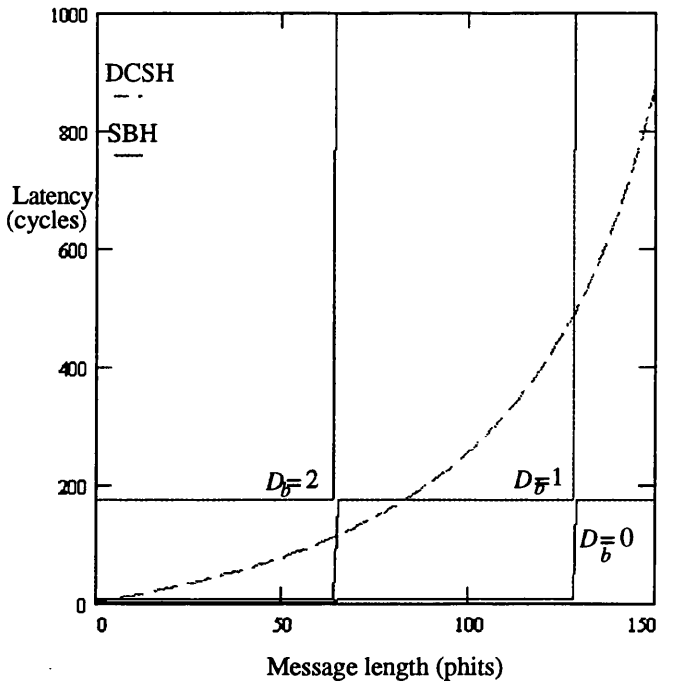



**Figure 4.6: Virtual-cut through performance in the SBH & DCSH for a fixed generation rate ($m$=0.006 message/cycle).**

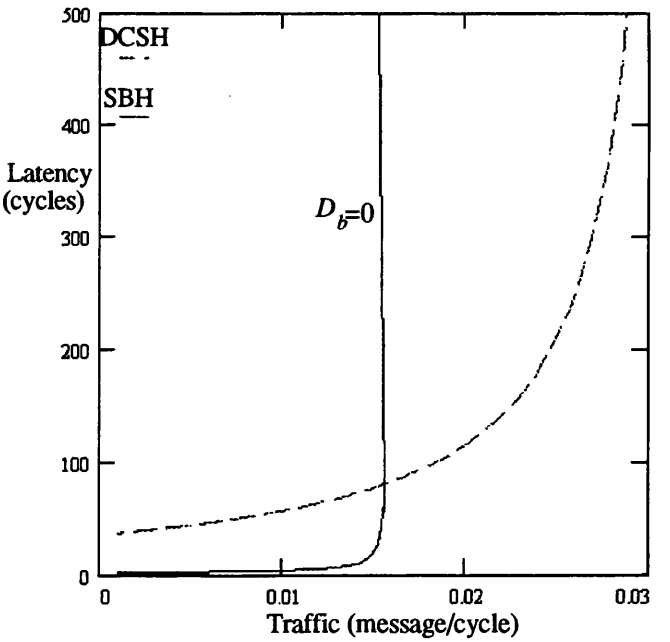



**Figure 4.7: Virtual-cut through performance in the SBH & DCSH for a fixed message length ($M$=32 phits).**

## Wormhole routing performance

With WR, the SBH outperforms the DCSH, as shown in Figure 4.8. Message blocking in WR increases quadratically with message length. Message blocking in the DCSH is higher since its message aspect ratio is much larger. Even when the bus-arbitration and release times are increased, the SBH still has better performance at long message length.

Figure 4.9, which shows message delays for different message lengths, reveals that as the bus-arbitration and release times are increased, the message length at which the DCSH overtakes the SBH increases. For example, for $D_b$=1, the DCSH is better when the message length is 16 phits (Figure 4.10). When $D_b$=2, the DCSH manages to provide a lower latency at 32 phits under high traffic (Figure 4.11). It can also be seen in Figure 4.11 that the DCSH has a clear advantage over the SBH at moderate and high traffic when $D_b$=3. This is because a short message length reduces the difference between the message aspect ratios in the SBH and DCSH relative to that mandated by the wiring density constraint. Further, the release and arbitration times have the same effect as increasing the message length in the SBH. These factors together makes the DCSH more favourable at short messages, which, as stated before, are typical of fine-grain computations [22, 38, 51, 52]; this also applies when technology offers high wiring density or pin-out counts; making network channels so wide that the message aspect ratio becomes very short.

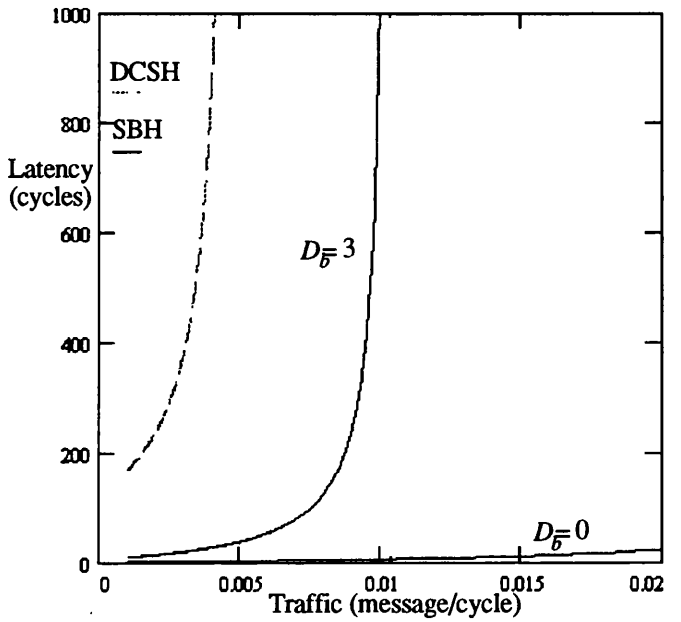



**Figure 4.8: Wormhole routing performance in the SBH & DCSH for a fixed message length ($M$=128 phits).**

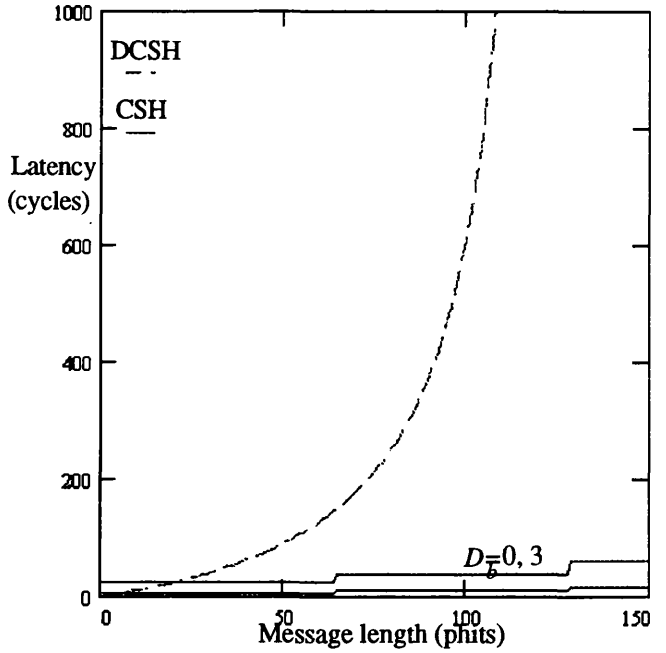



**Figure 4.9: Wormhole routing performance in the SBH & DCSH for a fixed generation rate ($m$=0.003 message/cycle).**

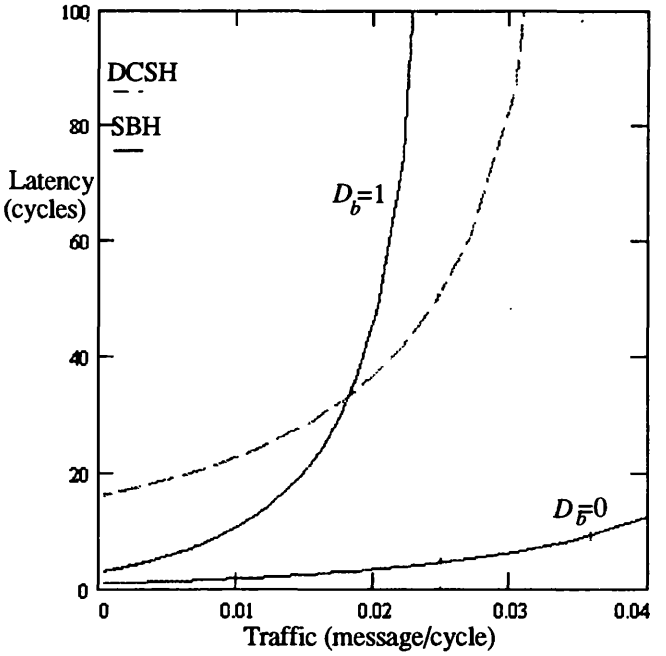



**Figure 4.10: Wormhole routing performance in the SBH & DCSH for a fixed message length ($M$=16 phits).**

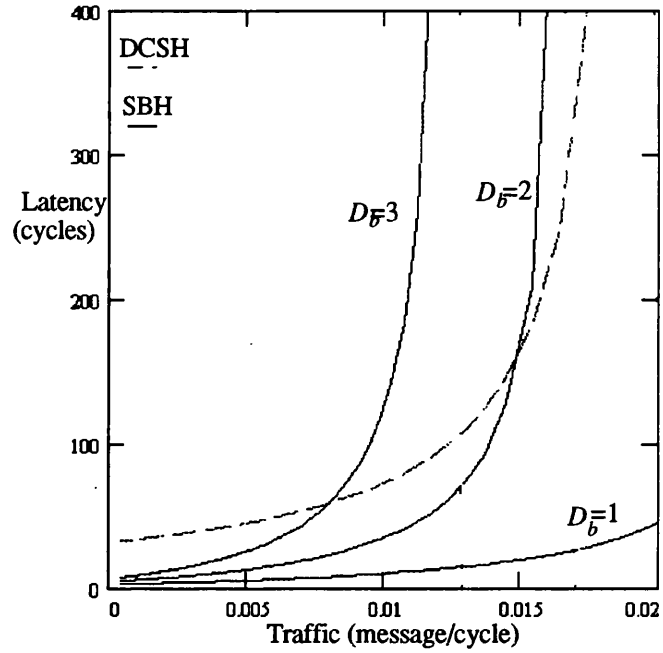



**Figure 4.11: Wormhole routing performance in the SBH & DCSH for a fixed message length ($M$=32 phits).**

## Pin-out

When the network is laid out over multiple chips, the number of pins is a limiting factor in network performance [23, 34, 49]. Assuming a node is implemented on a single chip, the pin-out in this case is the node degree times the channel width.

Let a bus in a $k^n$ node SBH have a width of $W_{SBH}$. Since a node can be either transmitting or receiving on a bus at any time but not both, one channel per node is sufficient to interface to the bus. Assuming that such a interface channel has the same width as the bus, the pint-out is

$$P_{SBH} = nW_{SBH} \tag{4.14}$$

Similarly, let the DCSH have a channel width of $W_{DCSH}$. The pin-out can be written as

$$P_{DCSH} = nkW_{DCSH} \tag{4.15}$$

To preserve the same pin-out in both networks, $W_{SBH}$ is then

$$W_{SBH} = kW_{DCSH} \tag{4.16}$$

This is the same channel width ratio that was found under the constant wiring density constraint, so the results of the previous section still apply.

### 4.2.3 Merits of the layered implementation

Practical bus systems usually suffer from bandwidth limitations as the system size scales. Some researchers have proposed the use of optical technology, which offers tremendous bandwidth, to relax this problem [88, 87, 89, 91]. However, the successful use of this technology cannot become a reality in parallel systems until problems related to opto-electronic conversion are overcome [112].

The SBH can also use the layered implementation presented in Chapter 2 to escape from the wiring density and pin-out constraints. However, there is a limit on how wide the bus can be due to difficulty in providing the same relay bandwidth inside its SEs, as pointed out by Athas [56]. The DCSH overcomes this problem by the inherent multiplexing-down effects in its SEs, which reduces the required relay bandwidth; only that of a single cluster channel is required, far less than for the SBH. This following results reveal to what extent the DCSH has to exploit the layered implementation to outperform the SBH.

#### Virtual cut-through performance

When the ratio between channel widths in the DCSH and SBH is 1:63, (*c.f.*, 1:64 under wiring density and pin-out constraints, the SBH loses to the DCSH even if bus arbitration and release times are ignored (Figures 4.12 and 4.13).

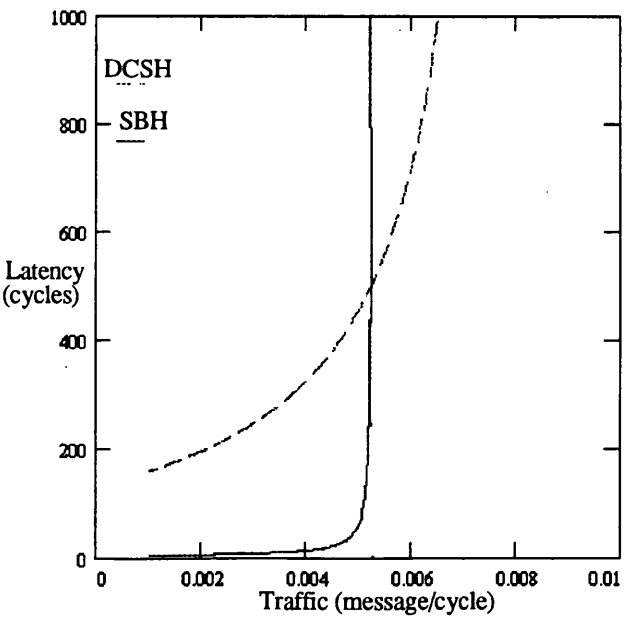



**Figure 4.12: Virtual-cut through performance in the SBH & DCSH for a fixed message length ($M$=128 phits), $W_{DCSH}$:$W_{SBH}$=1:63.**

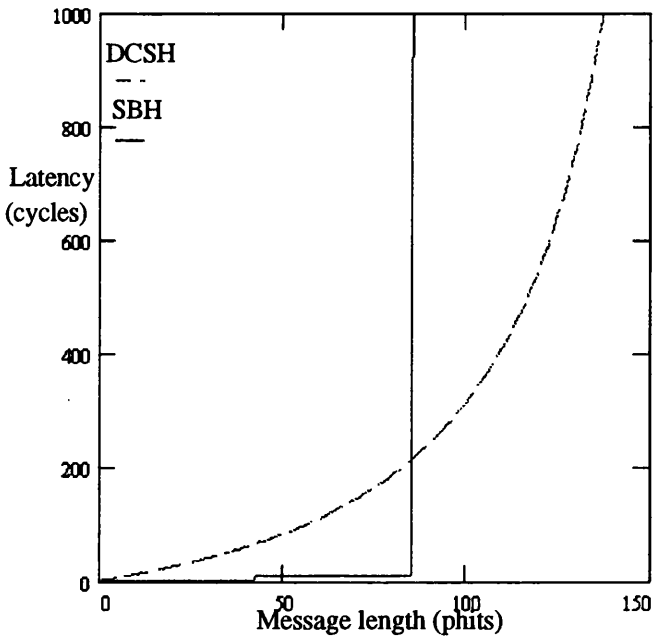



**Figure 4.13: Virtual-cut through performance in the SBH & DCSH for a fixed generation rate ($m$=0.006 message/cycle), $W_{DCSH}$:$W_{SBH}$=1:63.**

## Wormhole routing performance

When the channel width ratio between the DCSH and SBH is reduced to 1:8 both networks have a comparable latency under high traffic, even if bus-arbitration and release times are (unrealistically) ignored (Figures 4.14 and 4.15). When the ratio is decreased further to 1:7, the DCSH delivers lower latency at high and moderate traffic.

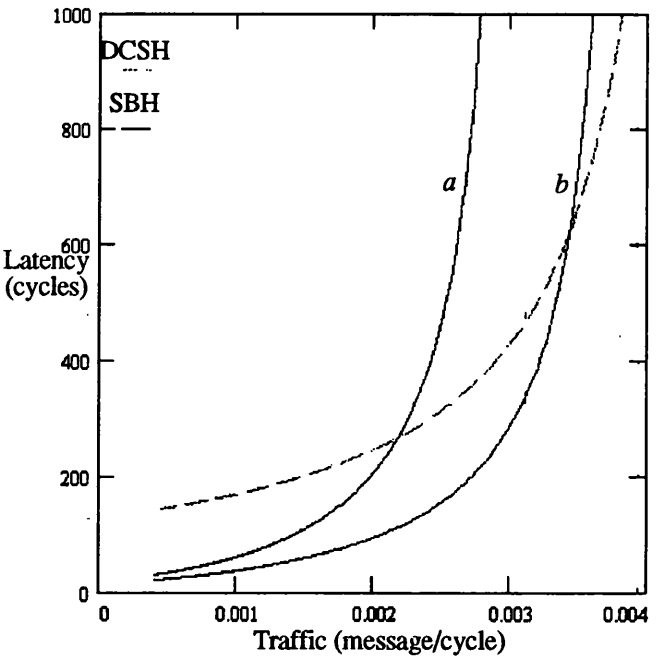



**Figure 4.14: Wormhole routing performance in the SBH & DCSH for a fixed message length (*M*=128 phits), *a*) $W_{DCSH}$:$W_{SBH}$=1:7, *b*) $W_{DCSH}$:$W_{SBH}$=1:8.**

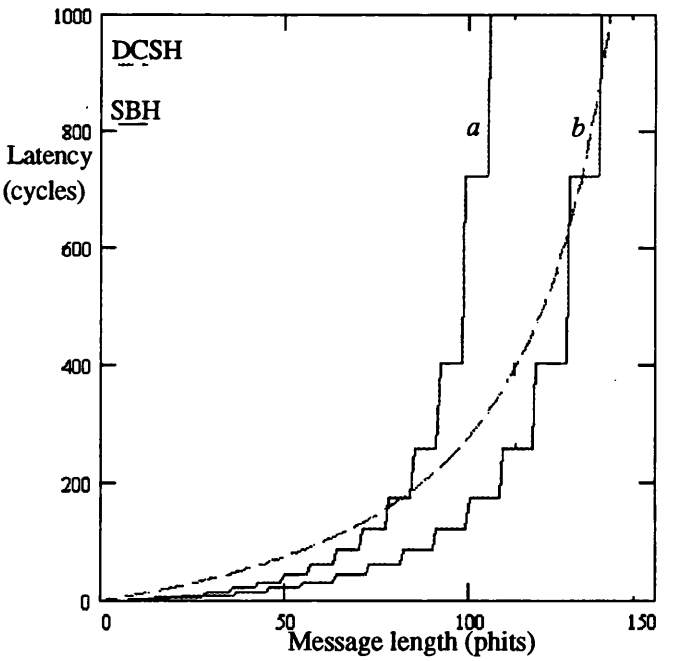



**Figure 4.15: Wormhole routing performance in the SBH & DCSH for a fixed generation rate (*m*=0.003 message/cycle), *a*) $W_{DCSH}$:$W_{SBH}$=1:7, *b*) $W_{DCSH}$:$W_{SBH}$=1:8.**

## 4.3 Crossbar switch hypermeshes (CSHs)

The cluster nodes in the CSH are connected by a crossbar switch. Recently, several researchers have suggested the use of crossbar switches because they offer a simpler way to implement hypermeshes [90, 92, 93, 102]. Such implementations cannot support large cluster sizes due to the limited sizes of commercially-available crossbar switches. The Prodigy and Genesis machines are based on this approach [90, 92].

Crossbar switch cost has often been measured in terms of the number of cross-points [84]. However, a crossbar switch is considered here to be implemented by means of multiple parallel channels (or buses) so as to be able to quantify the wiring density and pin-out of the CSH. As shown in Figure 4.16, the resulting structure is similar to that of the DCSH. The only difference is in the way these channels are used. In the CSH, each receiver node is allocated a channel for receiving messages from the other cluster nodes whereas in the DCSH each sender node is allocated a channel to send messages to all the cluster nodes.

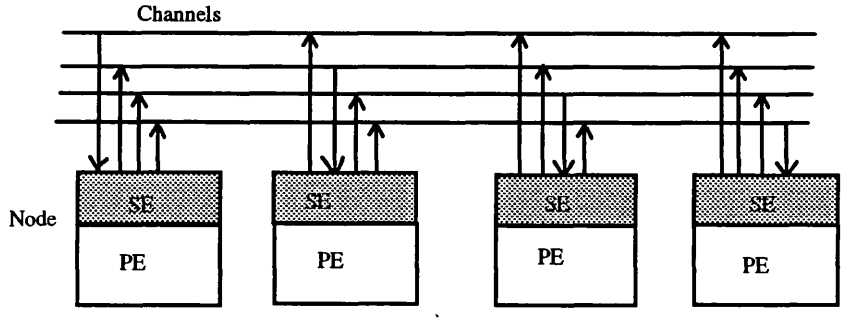



**Figure 4.16: A CSH cluster.**

### 4.3.1 Mathematical models for CSHs

The assumptions of Section 4.2.1 are used again. Only the necessary adjustments to the SBH model to adapt it for the CSH are shown. The analyses assume that a crossbar switch provides no message buffering. Designs which improve performance are possible, for example providing message buffering at either the input or output side of the crossbar switch [29]. However, the purpose of this section is to compare the DCSH implementation with that commonly described in the literature [90, 92, 93, 102].

## Virtual-cut through model

Hui and Arthurs have developed a queueing model for message switching in a $k \times k$ crossbar switch [131]. This model is adopted here for the CSH. The mean message response time, $r$ (i.e., the waiting time+service time), at a crossbar switch, with $\lambda$ messages/cycle arrival rate at each input channel and with each message requiring one cycle service time, is given by [131]

$$r = \frac{(2-\lambda)(1-\lambda)}{(2-\sqrt{2}+\lambda)(2+\sqrt{2}-\lambda)} \tag{4.17}$$

With VCT, only the mean message waiting time, $w$, has to be considered. Therefore, Equation 4.18 becomes

$$w = \frac{(2-\lambda)(1-\lambda)}{(2-\sqrt{2}+\lambda)(2+\sqrt{2}-\lambda)} - 1 \tag{4.18}$$

In the CSH, the rate of message traffic $\rho$ going out of each dimension is given as before by

$$\rho = \frac{m}{n} dB \tag{4.19}$$

($d$ is the same here as in the DCSH). The rate at an input channel of crossbar switch $i$ ($1 \le i \le n$) is $\rho$ messages /cycle and each message requires a $B$-cycle service time. Equation 4.19, after scaling the traffic rate and waiting times by a factor $B$, gives the waiting time of a $B$ phit message at crossbar switch $i$ as

$$w_o = \left(\frac{(2-\rho)(1-\rho)}{(2-\sqrt{2}+\rho)(2+\sqrt{2}-\rho)} - 1\right)B \tag{4.20}$$

## Wormhole routing model

Unlike in the SBH where a message has to compete for the single cluster bus, in the CSH a message has to compete for one of $k$ cluster buses. The traffic rate ($m_p$) that passes through dimension $i$ ($1 \le i \le n$) is the same as in the SBH. The traffic rate that is destined to a particular cluster node is therefore ($m_p / (k-1)$). Since there are ($k$-1) nodes which may send a message to the same node, a message sent to a particular cluster node sees, on average, $m_p$ other messages which are sent to the same node and hence use the same bus. Equation 3.31 then gives the latency seen by a message crossing dimension $i$ as

$$L_i^{CS} = \frac{m_p L_i^2}{2} = \frac{(1-\alpha)m' L_i^2}{2} \tag{4.21}$$

The latency seen by a message when entering dimension $i$ is similar to that of the SBH (Equation 4.9), except that bus contention is replaced by crossbar switch contention, and it can be written as

$$\begin{aligned} L_{i+1} = L + (1-\alpha)L_i^{CS} + \alpha^3(1-\alpha)m' L_i^2 \\ + \alpha(1-\alpha)^3 m'(L_i^{CS} + L_i)^2 \end{aligned} \tag{4.22}$$

### 4.3.2 Comparison of CSHs and DCSHs

As with the SBH, it is assumed that the channel width in the DCSH is one phit. Performance results are presented for $N$=4096 and MAM.

#### Wiring density

Since a crossbar switch in the CSH is assumed to be implemented by means of multiple buses, the wiring density is the same as for the DCSH, provided that their bus and channel widths are the same. Comparing both topologies under constant wiring density is therefore the same as comparing them assuming equal channel width.

#### Virtual-cut through performance

With VCT, the DCSH outperforms the CSH under moderate and high traffic, as shown in Figures 4.17 and 4.18. Providing more message buffering at the input and output sides of an SE allows the DCSH to sustain higher traffic rates.

The results of Section 3.4.1 have shown that in lower-dimensional DCSHs, ($n \leq 4$), the crossbar switch with input-only buffering has almost the same performance as that with both input and output buffering. Therefore, low-dimensional DCSHs with SEs which provide input-only buffering will yield the same results as those in Figures 4.17 and 4.18.

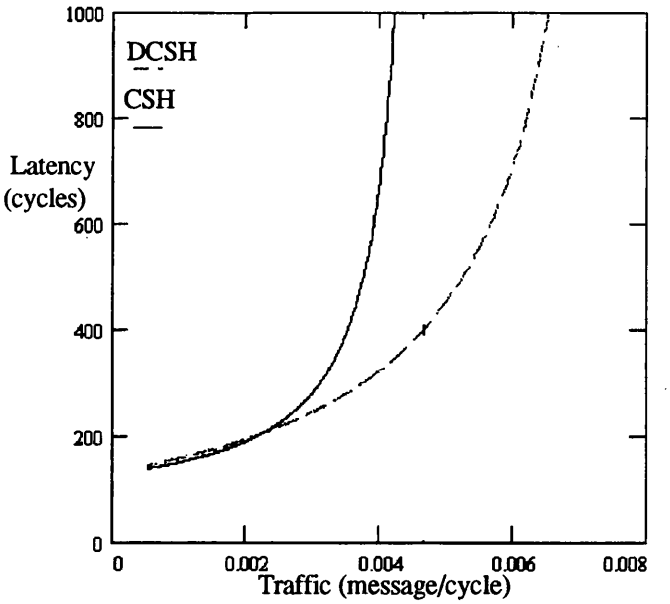



**Figure 4.17: Virtual cut-through performance in the CSH & DCSH for a fixed message length ($M$=128 phits).**

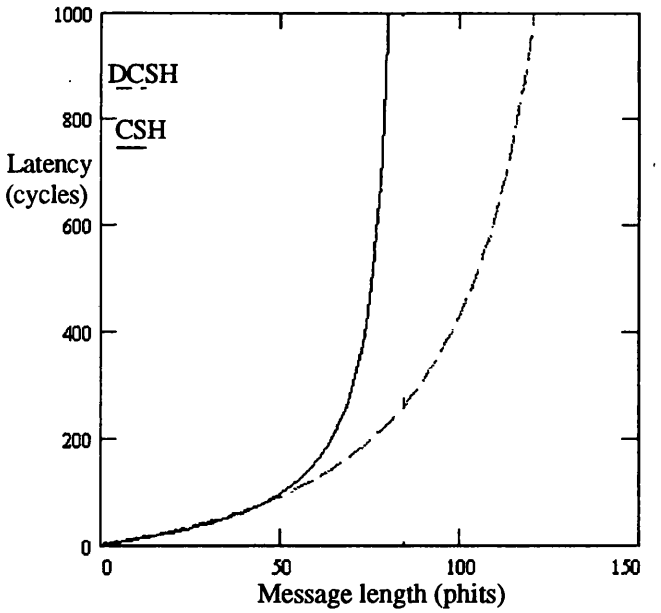



**Figure 4.18: Virtual cut-through performance in the CSH & DCSH for a fixed generation rate ($m$=0.006 message/cycle).**

## Wormhole routing performance

The message delays in the CSH and DCSH are identical (Figures 4.19 and 4.20). Having a channel for each sender (as in the DCSH) or a bus for each receiver (as in the CSH) has no affect on latency since the blocking behaviour in both structures is the same.

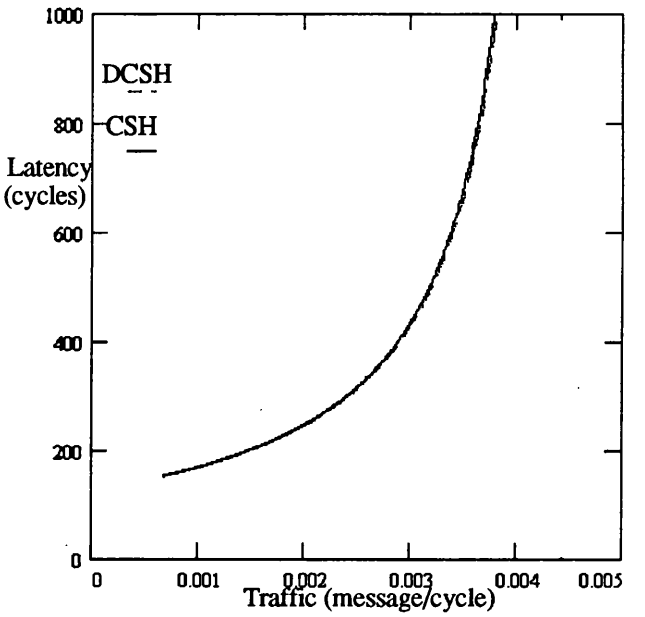



**Figure 4.19: Wormhole routing performance in the CSH & DCSH for a fixed message length ($M$=128 phits).**

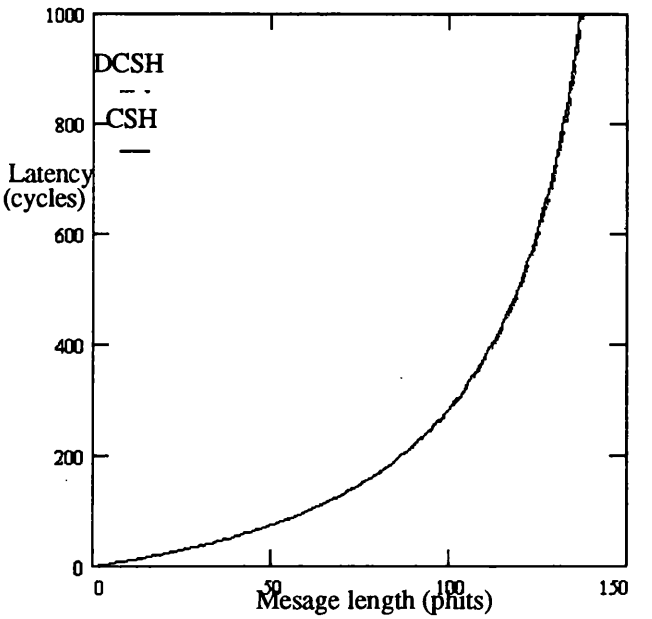



**Figure 4.20: Wormhole routing performance in the SBH & DCSH for a fixed generation rate ($m$=0.003 message/cycle).**

### Pin-out

Given that a bus in the CSH has a width of $W_{CSH}$, the pin-out can be written as

$$P_{CSH} = nkW_{CSH} \tag{4.23}$$

The pin-out in the DCSH is given by Equation 4.15. Both Equations 4.23 and 4.15 reveal that under constant pin-out, both the DCSH and CSH have the same channel width. Comparing the CSH to DCSH under constant pin-out is therefore the same as comparing them under constant wiring density constraint, so the results of the previous section still apply.

### 4.3.3 Merits of the layered implementation

The CSH cannot exploit the new implementation as its crossbar switches are inherently separate and independent units from its SEs. The DCSH, however, can use the layered implementation to accommodate more flit buffers and have wider channels than the CSH. The previous sections revealed that there is no difference in performance between the DCSH and CSH when WR is used, since both networks provide the same buffering capacity and have the same channel width. However, the DCSH can have a lower message aspect ratio than the CSH by exploiting the new implementation scheme (more buffering capacity has the same effect as increasing the channel width). As a result, message flits will occupy less network channels when they cannot advance through the network, reducing message blocking and hence latency.

## 4.4 Generalised hypercubes (GHs)

Cluster nodes in the GH are connected by means of a compete connection [109, 110]. The B-HIVE is an experimental machine which is based on this network [132, 133]. The GH is costly to implement since a node has a dedicated channel for every other cluster node. The DCSH requires less wiring density and pin-out because a node uses only a single channel to connect it to every other cluster node. With additional cost, however, the number of channels in the DCSH can be increased to any number. When a node has a dedicated channel for every other cluster node, the DCSH generalises to the GH. When $k$=2 in the GH, the binary $n$-cube is obtained which also belongs to the DCSH family.

### 4.4.1 Mathematical models for GHs

The following analyses use the assumption of Section 4.2.1.

#### Virtual-cut through model

Since a GH node has $n(k\text{-}1)$ network channels and a message travels on average $d$ hops ($d$ is the same as in the DCSH), the traffic rate on a network channel is given by

$$\rho = \frac{m}{n(k-1)} dB \tag{4.24}$$

Equation 3.3 gives the mean message waiting time at an output queue associated with a network channel as

$$w_o = \frac{(1-P_t)P_t B\rho}{(1-\rho)} \tag{4.25}$$

With SAM, the mean waiting time at the output queue leading to the PE is found to be

$$w_l = \frac{(2n(k-1)-1)P_t\rho}{2(1-n(k-1)P_t\rho)} B \tag{4.26}$$

#### Wormhole routing model

Only the modifications to the SBH model to adapt it for the GH are presented. Since a node in the GH has ($k$-1) incoming channels at dimension $i$, the traffic rate on one of these channels is given by

$$m_c = \frac{m'}{k-1} \tag{4.27}$$

The latency seen by a message entering dimension $i$ originating from input channel $j$ and crossing output channel $j$, can be computed. As depicted in Figure 4.21, the traffic rates of messages which originate from input channel $j$ at dimension $i$+1 and either pass or skip output channel $j$ at dimension $i$, is given by the following equations (the subscript $p$=pass, $s$=skip; the first subscript is for dimension $i$+1, and the second for $i$).

$$m_{sp} = m_{ps} = \frac{\alpha(1-\alpha)}{(k-1)^2} m_c \tag{4.28}$$

$$m_{pp} = \frac{(1-\alpha)^2}{(k-1)^2} m_c \tag{4.29}$$

$$m_{ss} = \frac{\alpha^2}{(k-1)^2} m_c \tag{4.30}$$

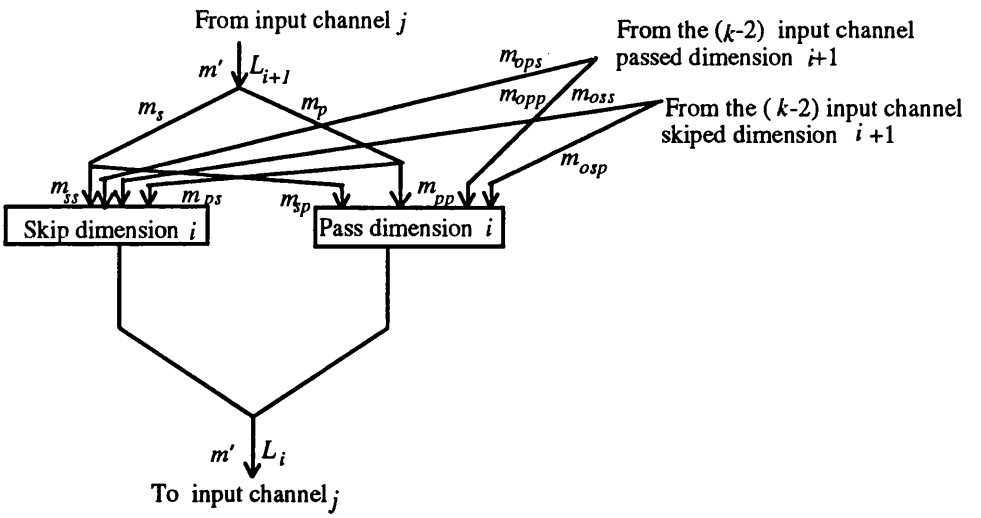



**Figure 4.21: A model to determine latency in a GH dimension.**

Figure 4.21 also shows that there is traffic which originates from the remaining ($k$-2) input channels and which may cross output channel $j$ at dimension $i$. This traffic consists of components that either pass or skip the dimension. Their rates are given by the following equations (the third subscript $o$=originated from the other ($k$-2) channels).

$$m_{spo} = m_{pso} = \frac{(k-2)\alpha(1-\alpha)}{(k-1)^2} m_c \tag{4.31}$$

$$m_{ppo} = \frac{(k-2)(1-\alpha)^2}{(k-1)^2} m_c \tag{4.32}$$

$$m_{sso} = \frac{(k-2)\alpha^2}{(k-1)^2} m_c \tag{4.33}$$

The latency seen by a message when entering dimension $i$, after adding contention as well as the latency due to routing at the lower dimensions with their appropriate weight, can be written as

$$L_{i+1} = L_i + \alpha(1-\alpha)^3 m_c L_i^2 + \alpha^3(1-\alpha) m_c L_i^2 \tag{4.34}$$

## 4.4.2 Comparison of GHs and DCSHs

Results are presented for $N$=4096. Further, both MAM and SAM are considered.

## Wiring density

In a $k^n$ node GH, every node in a row has $k/2$ channels crossing the midpoint of the network and are associated with the highest dimension. The bisection width of a GH with channel width of $W_{GH}$ is therefore given by

$$W_{GH} = 2\sqrt{N}\frac{\sqrt{N}}{2}\frac{k}{2}W_{GH} = \frac{Nk}{2}W_{GH} \tag{4.35}$$

The bisection width of an equivalent DCSH is given by Equation 4.12. To keep the same wiring density in both networks, $W_{GH}$ should be

$$W_{GH} = \left\lceil \frac{2}{k} W_{DCSH} \right\rceil \tag{4.36}$$

### Virtual-cut-through performance

With MAM, the DCSH has the optimal performance under light and moderate traffic because of its lower message aspect ratio, as shown in Figure 4.22. However, under high traffic the GH takes over. The expressions giving the traffic rate on network channels in the GH (Equation 3.40) and the DCSH (Equation 3.13) reveal that the traffic rate in former is $k/2(k\text{-}1)$ lower even though its message aspect ratio is $(k/2)$ larger. As a result, under high traffic the mean message waiting time at an output queue associated with a network channel in the GH is lower than that in the DCSH. Further, the GH is able to sustain a higher traffic rate. With SAM, the result for the GH are not shown in Figure 4.22 as it saturates at an earlier traffic rate. The DCSH then has a lower latency at all traffic conditions. This is because the large message aspect ratio in the GH increases blocking at the output queue leading to the PE.

Figure 4.23 shows that with both SAM and MAM, the DCSH has a lower latency when the message length is less than 135 phits since the traffic rate in the DCSH and GH is relatively light allowing the former to benefit from its wider channels. When the message length increases, the GH can cope better with the increased traffic.

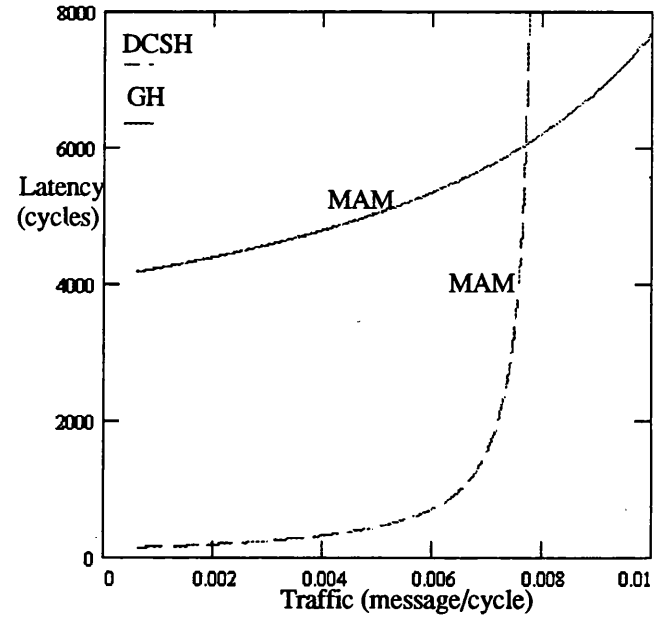



**Figure 4.22: Virtual-cut through performance in the GH & DCSH for a fixed message length ($M$=128 phits)[6].**

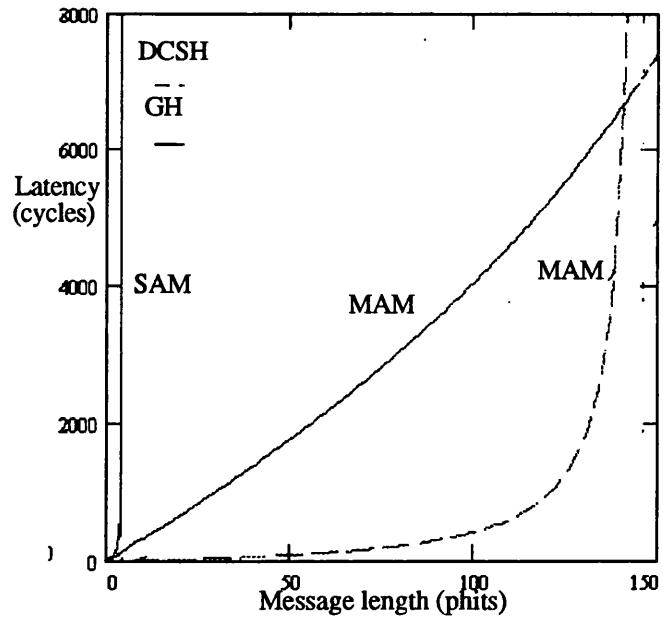



**Figure 4.23: Virtual-cut through performance in the GH & DCSH for a fixed generation rate ($m$=0.006 message/cycle)[6].**

[6] The GH is not shown for SAM because it saturates very early. Further, there is one trace for the DCSH as the results for MAM & SAM are the same.

### Wormhole routing performance

Results for different injection rates are omitted here because the GH saturates at a much earlier rate than the DCSH due to its large message aspect ratio. Figure 4.24 which shows performance results for different message lengths, reveals that the DCSH has better performance because of its wider channels.

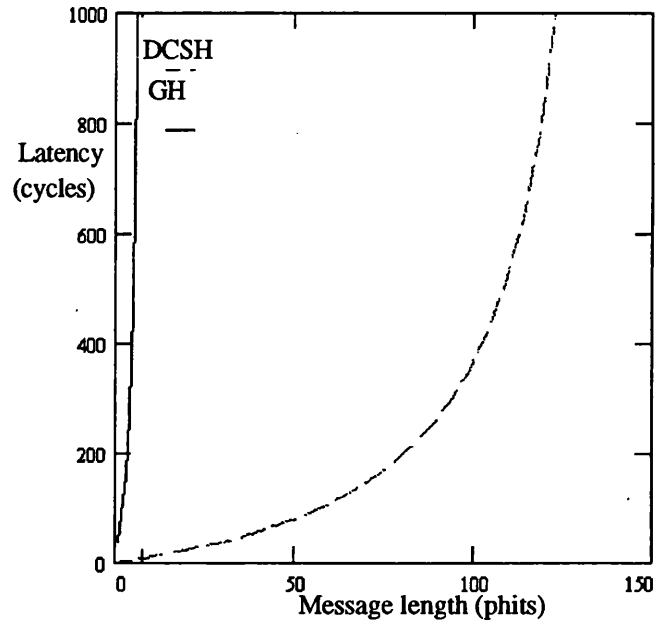



**Figure 4.24: Wormhole routing performance in the GH & DCSH for a fixed generation rate ($m$=0.003 message/cycle).**

## Pin-out

A node in the GH has ($k$-1) input and ($k$-1) network channels per dimension. The pin-out of the GH is therefore given by

$$P_{GH} = 2n(k-1)W_{GH} \tag{4.37}$$

In an equivalent DCSH, the pin-out is given by Equation 4.15. Under the constant pin-out constraint, the following should be satisfied

$$W_{GH} = \left\lceil \frac{k}{2(k-1)} W_{DCSH} \right\rceil \tag{4.38}$$

Under constant pin-out, the ratio between the channel widths in the GH and DCSH is smaller than under constant wiring density. However, since the general conclusions are

similar to those under the previous constraint, performance results under this constraint are not presented here.

### 4.4.3 Unconstrained implementation

This section compares the GH to the DCSH assuming no constraint on channel width. It evaluates the relative merits of using SAM and MAM in both the GH and DCSH, and also assesses the cost-effectiveness of using the complete connection of the GH.

#### Single versus multiple accepting model

With MAM, the GH outperforms the DCSH because the traffic rate in the former is lower (Figure 4.25 and 4.26). With SAM, on the other hand, the GH has a lower latency under moderate traffic. The GH, however, cannot sustain higher traffic than the DCSH despite its rich connectivity. The bottleneck at the output queue leading to the local PE makes the GH saturate at the same traffic rate as the DCSH does. The only way for the GH to exploit its rich connectivity is to remove this bottleneck by using MAM.

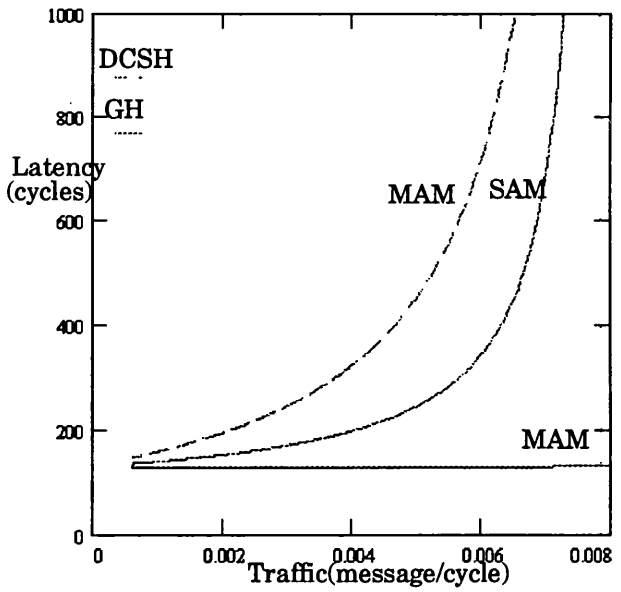



**Figure 4.25: Virtual-cut through performance in the GH & DCSH for a fixed message length ($M$=128 phits)[7].**

[7] There is one trace for the DCSH because the results for MAM and SAM are the same.

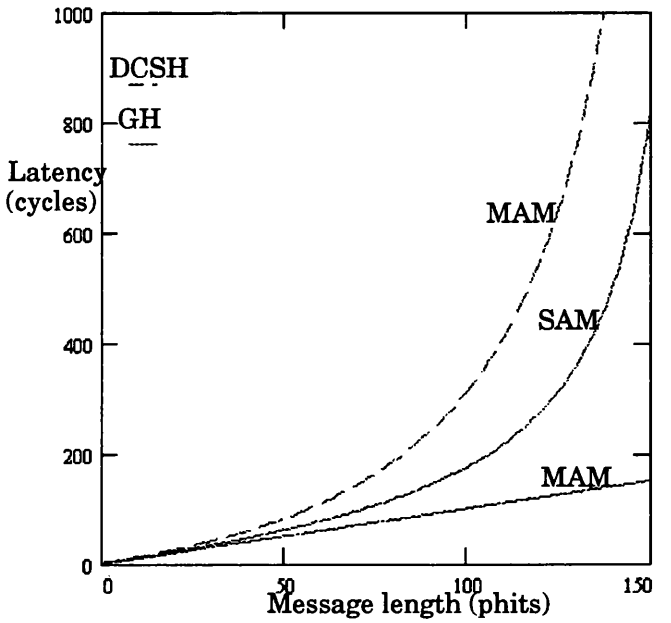



**Figure 4.26: Virtual-cut through performance in the GH & DCSH for a fixed generation rate ($m$=0.006 message/cycle)[7].**

## Cost-effectiveness of the complete connection

The GH is compared to the symmetric split-DCSH (SS-DCSH), which is a variation of the DCSH, in order to test the cost-effectiveness of the complete-connection scheme in the former. The SS-DCSH is less costly to implement than the GH as its nodes have only two channels per cluster, where one channel is used to communicate with even-numbered nodes and the other with odd-numbered ones (Section 3.6 presents an analysis of the AS-DCSH).

With MAM, the GH has much better performance than the SS-DCSH at moderate and high traffic (Figure 4.27 and 4.28). With SAM, the connection scheme of the SS-DCSH can deliver similar performance to that of the complete-connection. This is because the traffic rate in the SS-DCSH is reduced by half, compared to that in the DCSH. It can therefore be concluded that the use of complete-connection is not cost-effective. The only way for the GH to take advantage of its rich connectivity is to use MAM and this would increase its implementation cost.

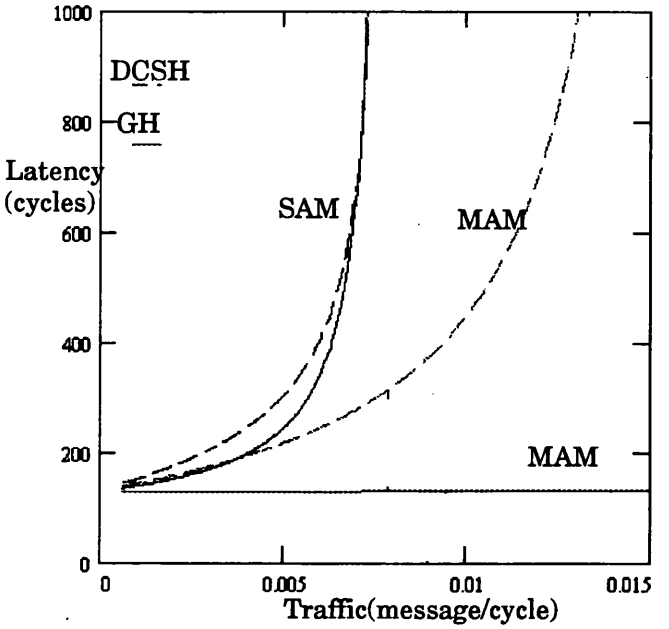



**Figure 4.27: Virtual-cut through performance in the GH & SS-DCSH for a fixed message length ($M$=128 phits).**

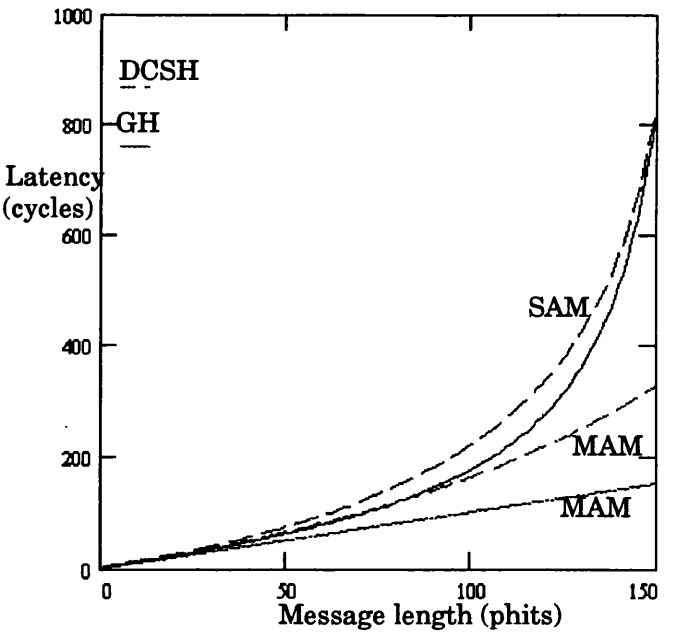



**Figure 4.28: Virtual-cut through performance in the GH & SS-DCSH for a fixed generation rate ($m$=0.006 message/cycle).**

## 4.5 Conclusions

This chapter has evaluated the performance merits of the DCSH over the SBH, CSH, and GH assuming the same implementation cost in VLSI and multiple-chip technology. (Although the results have been shown for the low-dimensional cases, the conclusions drawn here can also be applied to higher-dimensional cases.)

With virtual cut-through, when the bus arbitration and release times are included, the DCSH outperforms the SBH under moderate and high traffic. Further, the DCSH can sustain a higher traffic load. With the unrealistic assumption of zero bus-arbitration and release times, the SBH has a lower latency under light and moderate traffic. However, even though a bus in the SBH is much wider than a channel in the DCSH, the two structures still saturate at the same traffic rate because of high bus contention in the former network.

When the message aspect ratio is less than the bus width in the SBH, the DCSH provides superior performance. A message aspect ratio less than the bus width may be encountered in a SBH in two situations: first, where high wiring density or pin-out counts are available; second, where network sizes are increased and the message length is kept constant, the SBH bus becomes relatively wider than a DCSH channel. In these situations, the DCSH uses its channel width more efficiently.

With wormhole routing, the SBH outperforms the DCSH at long message lengths. But, as the message length is decreased, the DCSH becomes more favourable. The benefits of the DCSH are more apparent as the bus-arbitration and release times increase.

With virtual cut-through, the DCSH provides better performance than the CSH because of the provision of more message buffering in the former. With wormhole routing, the DCSH and CSH have comparable performance since whether a crossbar switch in a cluster is distributed (as in the DCSH) or not distributed (as in the CSH) has no impact on performance.

With both virtual cut-through and wormhole routing, the DCSH has better latency characteristics than the GH. Practical systems, such as the B-HIVE, which are based on the GH, have employed the single-accepting model [132, 133]. However, the GH must use the multiple-accepting model to take full advantage of its rich connectivity. The GH

is costly to implement both in VLSI and multiple-chip technology. The multiple-accepting model makes this network much less desirable as it further increases implementation cost.

However, if the layered implementation is employed, the DCSH always provides the optimal performance. The other hypermeshes are unable to take advantage of this implementation: the relay bandwidth constraint limits the SBH; the CSH cannot use the scheme since its crossbar switches are self-contained units, separate from its switching elements; while the use of complete connection in the GH prevents it from taking advantage of the layered implementation altogether.

# 5 Performance comparison with meshes

## 5.1 Introduction

Meshes are a variation of bi-directional $k$-ary $n$-cubes where the wrap-around connections are omitted [3, 22]. In a $k^n$ node mesh, each of the $n$ dimensions contains $k$ nodes, each of which is connected only to its neighbours. The binary $n$-cube, which is a special case of the DCSH family, is also a special case of the mesh family with $k$=2.

As already discussed, Dally has shown that under the constant wiring density constraint, low-dimensional tori outperform high dimensional binary $n$-cubes [3, 22, 38]. His results have greatly influenced the design of multicomputers. Several current commercial systems are based on low-dimensional tori, and especially meshes. Examples of these systems include the Symult 2101 [44], Delta-Touchstone [48], CMU-Intel iWarp [45], CRAY T3D [134], and the MIT Alewife [135]. (In what follows the term "mesh" is used interchangeably with "low-dimensional mesh".)

The mesh has been more popular than the torus in real machine designs because it can use a simpler deadlock-avoidance algorithm [81, 82]; deadlock is simply avoided by

forcing messages to visit dimensions in a pre-defined order. As a result, the SEs are simple to implement and are fast [64, 76, 66]. Restricted routing, however, cannot ensure deadlock avoidance in the torus. Several virtual channels are required which are time-multiplexed into the physical channels [81, 82]. These virtual channels require extra buffering resources, resulting in more complex and therefore slower SEs [64, 76 66]. Furthermore, the mesh can be partitioned into smaller parts that are still meshes, which is desirable for some parallel applications [4].

Meshes are popular because of their perceived *modularity;* they can be expanded simply by adding new nodes and channels without any change to the existing node structure (e.g., pin-out). Since node chips can be used as elementary building blocks, they are potentially marketable components [64, 136]. Unfortunately, this modularity is at the expense of performance, for in a fixed-degree network, as the size grows the channels must be increased in bandwidth to maintain the same latency [13, 137]. The number of pins on a mesh node chip must therefore be increased with system size, a fact which is obvious in a topology like the DCSH or binary $n$-cube whose degree increases as the number of nodes grows, but less apparent for the mesh.

Meshes are particularly suitable for applications with nearest-neighbour communications such as finite element computations [84, 85]. As discussed in Chapter 2, some researchers have augmented the basic structure with buses so as to support applications that possess less locality and to efficiently perform important operations, such as broadcast which are found in several scientific calculations [105, 106]. However, DCSHs offer more powerful and richer interconnect structures. As well as being inherently suitable for mesh-like applications, they support broadcast operations efficiently [50, 103, 121]. Further, important common process graphs such as the binary tree and the FFT pattern can be mapped more successfully on DCSHs [87, 138].

Low-dimensional meshes map well into physical dimensions and therefore in theory can have wide channels [7, 22, 38]. However, their SEs have to provide comparable relay bandwidth to be able to take advantage of these wide channels. High relay bandwidth results in more costly and complex SE designs [56]. More recently, Athas has presented a layered implementation, at first sight similar to that for low-dimensional DCSHs, to increase the mesh channel width. However, the high relay bandwidth required at SEs renders the exploitation of high wiring densities or pin-out counts impractical. The relay bandwidth requirements in a DCSH are lower than that in a mesh with the same bisection width or pin-out due to the multiplexing-down inherent in the DCSH topology.

The DCSH outperforms the mesh in the absence of any constraint on channel width because of its lower diameter. This chapter, however, compares the two networks using the same strategy as in Chapter 4, i.e. considering implementations in VLSI and multiple-chip technology [22, 38, 23, 34] as well as those based on the layered implementation scheme [50, 103, 121].

## 5.2 Mathematical models for meshes

The following analyses assume that restricted routing in the $k^n$ node mesh is used. A node generates a message, with probability $m$ in a cycle, of a fixed length ($B$ phits) and destined to the ($N$-1) nodes with equal probability.

### Virtual cut-through model

Agarwal has derived a queueing model for a $k^n$ node mesh with the multiple-accepting (MAM) [52]. The model ignores the fact that a node does not reference itself. To include this restriction and adapt the model to the single-accepting model (SAM) is straightforward. The mean total message latency is given by

$$L = (D_t + w_o)nd + (1 + w_l) + B \tag{5.1}$$

where $D_t$ is the decision time, and $w_o$ and $w_l$ are the mean waiting times at an output queue associated with a network channel and a local channel respectively. The expression for $w_o$ can be written

$$w_o = \frac{\rho}{2(1-\rho)} \frac{(d/n-1)}{d^2/n^2} \frac{(n+1)}{n} B \tag{5.2}$$

where $\rho$ is the channel utilisation and it is given by

$$\rho = \frac{m}{2n} dB \tag{5.3}$$

$d$ is the average message distance and can be written as

$$d = \frac{n}{3}\left(k - \frac{1}{k}\right)\frac{N}{N-1} \tag{5.4}$$

Equation 3.14 gives the mean waiting at the output queue associated with the local channel as

$$w_l = \frac{P_t(n-1)B\rho}{2(1-nP_t\rho)} \tag{5.5}$$

where $P_t=1/d$ is the probability of termination [28, 34].

## Wormhole routing model

A wormhole routing model requires a little more effort. The approach is to adapt the model developed by Dally for unidirectional $k$-ary $n$-cubes, first with zero decision times, then with extensions to include the effects of higher decision times [22, 38].

Each of the two directions at dimension $i$ ($1 \le i \le n$) receives messages at a traffic rate of

$$m' = \frac{m}{2}\frac{N}{N-1} \tag{5.6}$$

As in Section 3.2.3, latency is computed starting from the lowest dimension 0 (i.e., destination node) and propagated back to the highest dimension $n$ (i.e., source node). The latency at dimension 0 is given by

$$L_0 = B \tag{5.7}$$

Without loss of generality, the following discussion focuses on the right direction at dimension $i$. Figure 5.1 shows that the traffic arriving at this direction is composed of four streams: a stream that skips dimension $i$+1 and comes from the right direction; a stream that skips dimension $i$+1 and comes from the left direction; a stream that passes through dimension $i$+1, and comes from the left direction; and a stream that passes through dimension $i$+1 and comes from the left direction. Their respective rates are given by (the subscript $p$=pass, $s$=skip, $r$=right, $l$=left)

$$m_{sr} = m_{sl} = \alpha\frac{m'}{2} \tag{5.8}$$

$$m_{pr} = m_{pl} = (1-\alpha)\frac{m'}{2} \tag{5.9}$$

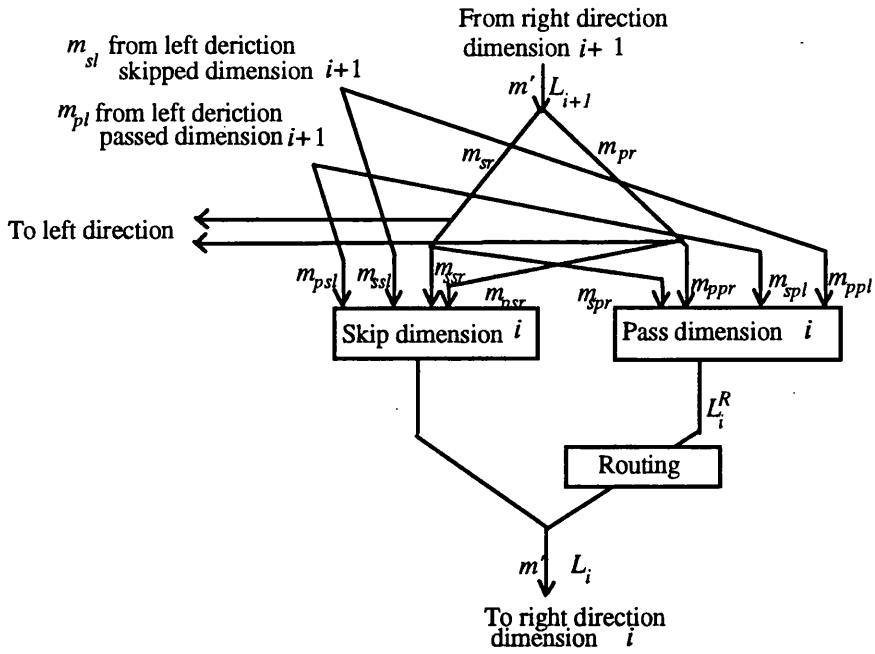



**Figure 5.1: A model to determine latency in a mesh dimension.**

where $\alpha=1/k$ is the probability that a message either passes or skips dimension $i$. Each of these four streams splits into two components: one passes through dimension $i$ and competes with components from the other streams to access the entering channel and encounters *routing* latency through this dimension, the other skips dimension $i$ and competes with components from the other streams for access to an entering channel, corresponding to another lower dimension. The different components, as shown in Figure 5.1, have the following traffic rates (the first subscript is for dimension $i$+1, the second for $i$, and the third for the direction).

$$m_{ppr} = m_{ppl} = (1-\alpha)^2 \frac{m'}{2} \tag{5.10}$$

$$m_{psr} = m_{psl} = \alpha(1-\alpha)\frac{m'}{2} \tag{5.11}$$

$$m_{ssr} = m_{ssl} = \alpha^2 \frac{m'}{2} \tag{5.12}$$

$$m_{spr} = m_{spl} = \alpha(1-\alpha)\frac{m'}{2} \tag{5.13}$$

The latency seen by a message entering rightwards at dimension $i$, after adding the different components with their appropriate weights, is found to be

$$L_{i+1} = L_i + (1-\alpha)L_i^R + \frac{(1-\alpha)^3(1+3\alpha)m'}{2}\frac{(L_i + L_i^R)^2}{2} + \frac{\alpha^2(1+2\alpha-2\alpha^2)m'}{2}\frac{L_i^2}{2} \tag{5.14}$$

The routing latency seen by a message at dimension $i$ is computed using the model depicted in Figure 5.2 [22, 38]. Each node along dimension $i$ handles two sources of traffic: entering traffic $m_r$ and continuing traffic $m_c$. Figure 5.1 and 5.2 reveal that messages enter rightwards at dimension $i$ with a rate of

$$m_r = m_{ppr} + m_{spr} + m_{ppl} + m_{spl} = (1-\alpha)m' \tag{5.15}$$

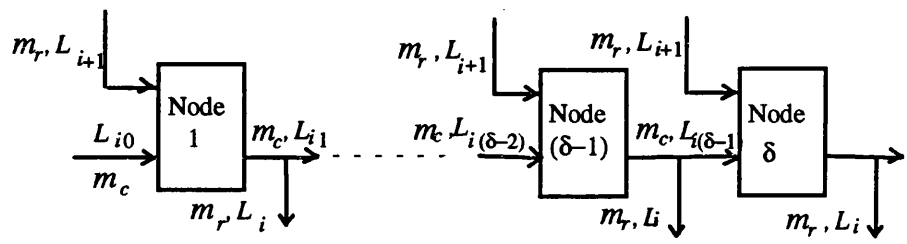



**Figure 5.2: A model to determine the routing latency.**

The average number of continuing channels that a message goes through at dimension $i$ is given by [52]

$$\delta = \frac{1}{3}(k - 1 - \frac{1}{k-1}) \tag{5.16}$$

The traffic on the continuing channels is

$$m_c = \delta m_r \tag{5.17}$$

Let $L_{ij}$ denote the latency seen by a message on the $j$+1st channel it encounters after arriving at dimension $i$. Latency on the $j^{th}$ continuing channel is given by

$$L_{ij} = L_{ij-1} + m_c \frac{L_{i0}^2}{2} \tag{5.18}$$

As shown in [22, 38], repeating this calculation $\delta$ times gives the latency seen by a message on the first entering continuing channel in term of itself as

$$L_{i0} = L_i + m_c \frac{L_{i0}^2}{2} \tag{5.19}$$

Solving this equation gives

$$L_{i0} = \frac{1 - \sqrt{1 - 2m_c L_i}}{m_c} \tag{5.20}$$

and, therefore, the routing latency at dimension $i$ is

$$L_i^R = L_{i0} + \frac{m_c}{2} L_{i0}^2 - L_i \tag{5.21}$$

The model can now be extended to include the effects of the decision time. Equation 5.7, giving the latency at the destination node (i.e. dimension 0), becomes

$$L_0' = D_t + B \tag{5.22}$$

Equation 5.18, giving the latency on the $j^{th}$ continuing channel due to contention at the successive $j$-$1^{th}$ continuing channel, can be written as

$$L_{ij}' = D_t + L_{ij-1}' + m_c \left[ \frac{L_{i0}'}{2} \right]^2 \tag{5.23}$$

After repeating this calculation $\delta$ times, Equation 5.20, which gives the latency seen by a message on the first entering continuing channel, becomes

$$L_{i0}' = \frac{1 - \sqrt{1 - 2m_c(\delta D_t + L_i')}}{m_c} \tag{5.24}$$

Equation 5.18 accounts only for the effects of the decision time at $\delta$ SEs when a message is routed through dimension $i$. However, to include the effect of decision time at the first SE of this dimension, equation 5.13 becomes

$$L_{i+1}' = D_t + L_i' + (1-\alpha)L_i^R + \frac{(1-\alpha)^3(1+3\alpha)m'}{2} \frac{(L_i' + L_i^R)^2}{2} + \frac{\alpha^2(1+2\alpha-2\alpha^2)m'}{2} \frac{L_i^2}{2} \tag{5.25}$$

Simulations have shown that the divergence between the above model and simulation increases as the traffic increases because the model ignores the edge effects of the mesh due to the absence of wrap-around connections. Nevertheless, the results produced by the model are sufficiently accurate to be used as a basis for comparing DCSHs to

meshes. The conclusions reached from the following comparisons are in close agreement with those provided by simulation [50, 121].

## 5.3 Comparison of meshes and DCSHs

The following discussion focuses on the 2-dimensional case since most current multicomputers are based on the 2-dimensional mesh. It is assumed that the channel width in the DCSH is one phit. Thus the message lengths which are considered here represent message aspect ratios for the DCSH. Performance results are presented for $N$=4096, which represents a moderately large system (other sizes are considered if they yield different results). For VCT, only results for zero-cycle decision times are shown since it has been found that increasing the decision time to more realistic values does not change the results. Further, the discussion is focused on MAM (results for SAM are shown if they are different).

### Wiring density

The bisection width of a mesh with a channel width of $W_{Mesh}$ is given by [4, 22]

$$W_{Mesh} = 2\sqrt{N}k^{n/2-1}W_{Mesh} \tag{5.26}$$

Section 4.2.3 gives the bisection width of an equivalent DCSH with a channel width $W_{DCSH}$ as

$$W_{DCSH} = NW_{DCSH} \tag{5.27}$$

In order to have the same wiring density in the mesh and DCSH, the channel width in the former should be

$$W_{Mesh} = \left\lceil \frac{k}{2} W_{DCSH} \right\rceil \tag{5.28}$$

### Virtual cut-through performance

Figure 5.3 shows the mean latency as a function of the generation rate in the mesh and DCSH when the message length is 128 phits, which is a typical length for fine-grain computation [22, 37, 51, 52]. The mesh outperforms the DCSH because its smaller message aspect ratio results in lower traffic rate on its network channels.

Figure 5.4 shows the effect of message length on system performance. When the

message length is smaller than the mesh channel width, the DCSH outperforms the mesh. Figure 5.5 shows the latency in both systems when $M$=8 phits. Although the channel width ratio of the two topologies is 32, the effective ratio is less than this and equal the message length (due to the short message length). As a result, the mesh cannot use its channel bandwidth efficiently. When the message length is greater than the mesh channel width, the mesh has a lower latency because the effective channel width ratio is large enough to allow it to outperform the DCSH. (The results for SAM and MAM in the mesh are the same, since the bottleneck resides at the output queue associated with the network channel rather than at the queue leading to the local PE).

The above results reveal that the DCSH is more efficient in exploiting very wide channels, because the channel width in the mesh may be larger than the message length, in which case some bandwidth is wasted. This problem might be encountered in the DCSH only when wiring density is extremely high [50, 53]. Further, as the network size increases, and the message length is kept constant, the message aspect ratio of the mesh eventually becomes smaller than the channel width. In such a situation, the DCSH outperforms the mesh even though the latter has a much wider channel.

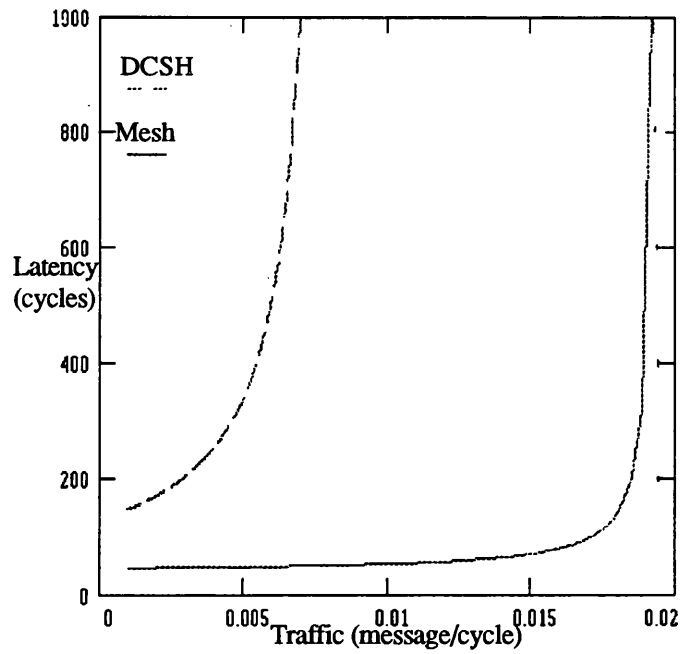



**Figure 5.3: Virtual cut-through performance in the mesh & DCSH for a fixed message length ($M$=128 phits).**

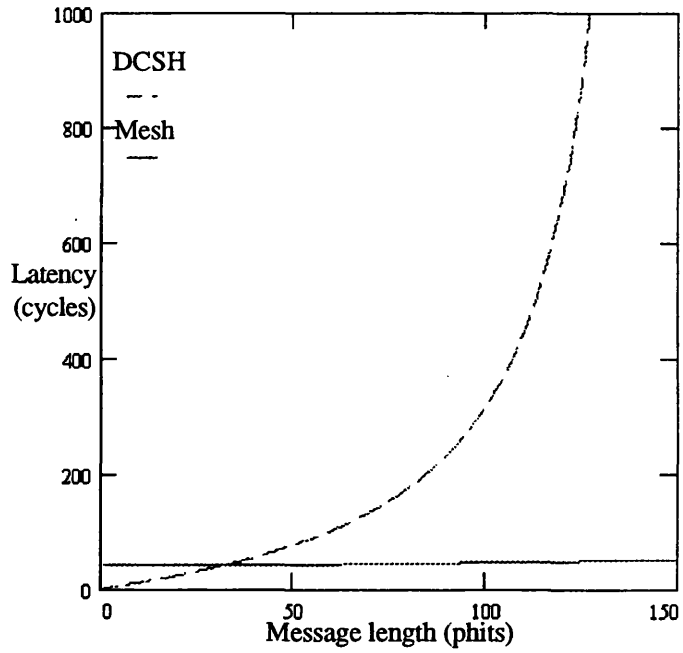



**Figure 5.4: Virtual cut-through performance in the mesh & DCSH for a fixed generation rate ($m$=0.005 message/cycle).**

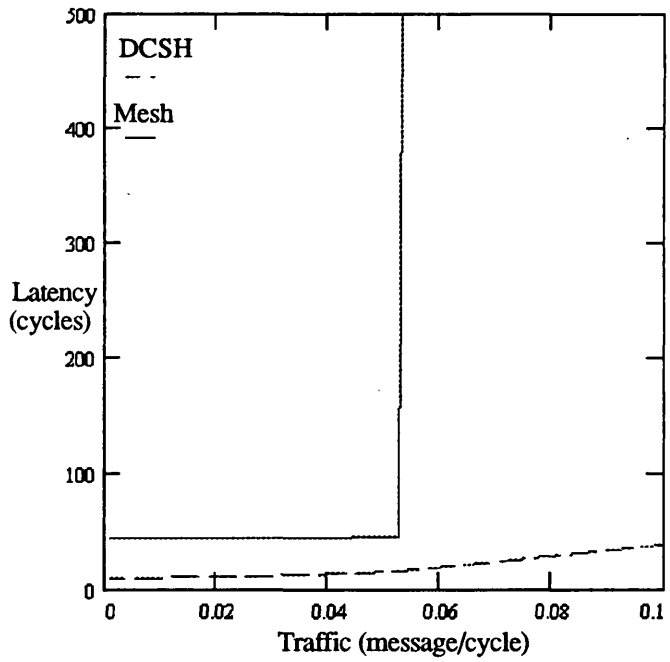



**Figure 5.5: Virtual cut-through performance in the mesh & DCSH for a fixed message length ($M$=8 phits).**

### Wormhole routing performance

However, when the decision time is only one cycle, which underestimates current figures [2, 4, 65, 66, 76], the DCSH has a lower latency than the mesh under moderate and high traffic, as depicted in Figure 5.6 and 5.7. The mesh can exploit its wider channels under light traffic only. Message blocking in WR increases quadratically with the message length. Increasing the decision time has the same effect as increasing the message length, which results in quadratic increase in message blocking. As decision time increases, the DCSH performance degrades slightly. The mesh performance, however, suffers considerable degradation and thus saturates at a much earlier traffic rate. This is because a message in the mesh, on average, crosses a large number of SEs and therefore requires a long service time to reach its destination. Figures 5.6 and 5.7 also reveal that when the decision time is ignored, which is unrealistic given current technology, the mesh performs better than its DCSH equivalent under all traffic conditions because its wider channels reduces message blocking.

When the network size is small (e.g., $N$=$16^2$), the DCSH cannot outperform the mesh unless the decision time is increased beyond 2 cycles, as revealed in Figure 5.8 and 5.9. When the decision time is $D_t$=1 or 2 cycles, the $16^2$ node DCSH has lower performance than its mesh counterpart. This is due to fact that for a small network size, a message in a mesh has to go across a less number of SEs, and thus needs a lower service time to cross the network, resulting in lower message blocking. When the decision time increases to 3 cycles, the mesh performance degrades and the DCSH becomes more favourable.

In current practice and for the foreseeable future, the decision time must be carefully considered in multicomputer designs. Even though the mesh can have wider channels than the DCSH, its extreme sensitivity to decision time offsets the benefits of its wider channels. As the mesh system size increases, its performance degrades sharply even when the decision time is unrealistically low. The DCSH has an important advantage over the mesh in that it is almost insensitive to this delay factor. This is because a message crosses only a few SEs along its journey. It has been found that even when the decision time in the DCSH is increased to unrealistic values (e.g., 10 cycles) the DCSH still provides better performance. It therefore can be concluded that the DCSH is more favourable than the mesh when WR is used because it can use slower (and therefore less costly) SEs and still deliver the optimal performance.

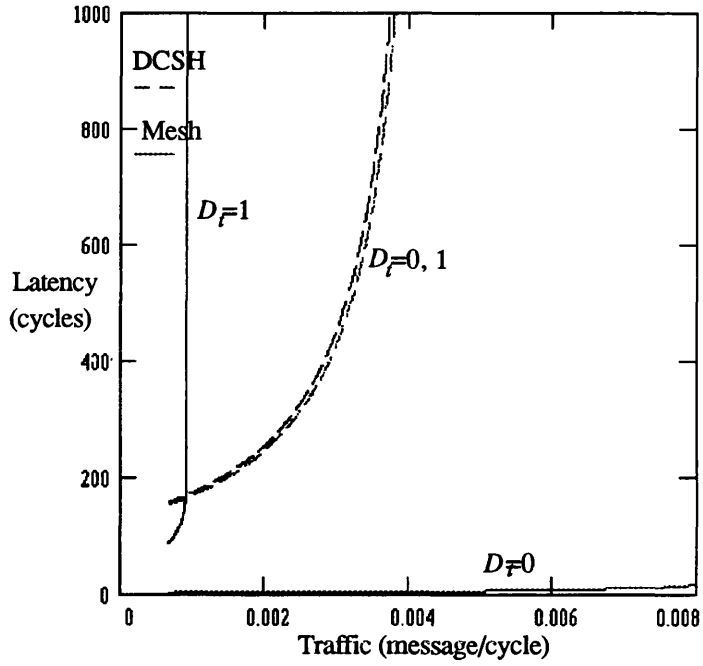



**Figure 5.6: Wormhole routing performance in the mesh & DCSH for a fixed message length ($M$=128 phits), $N$=4096.**

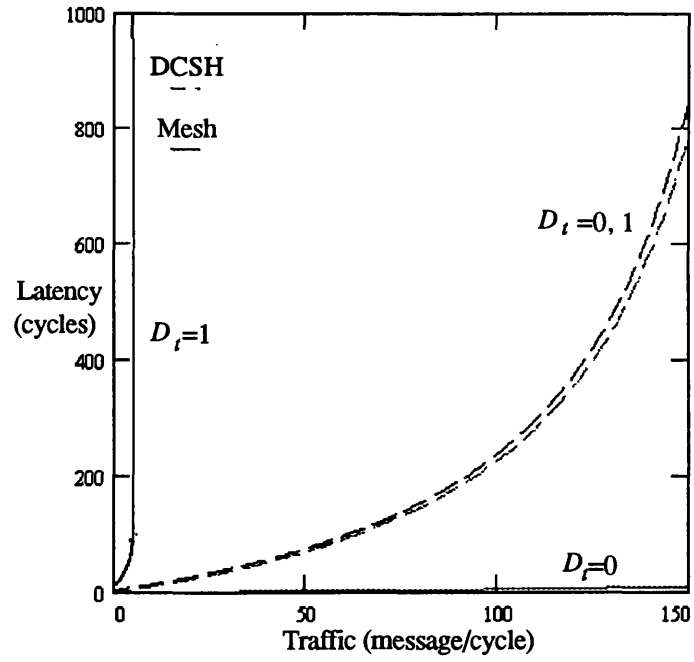



**Figure 5.7: Wormhole routing performance in the mesh & DCSH for a fixed generation rate ($m$=0.003 message/cycle), $N$=4096.**

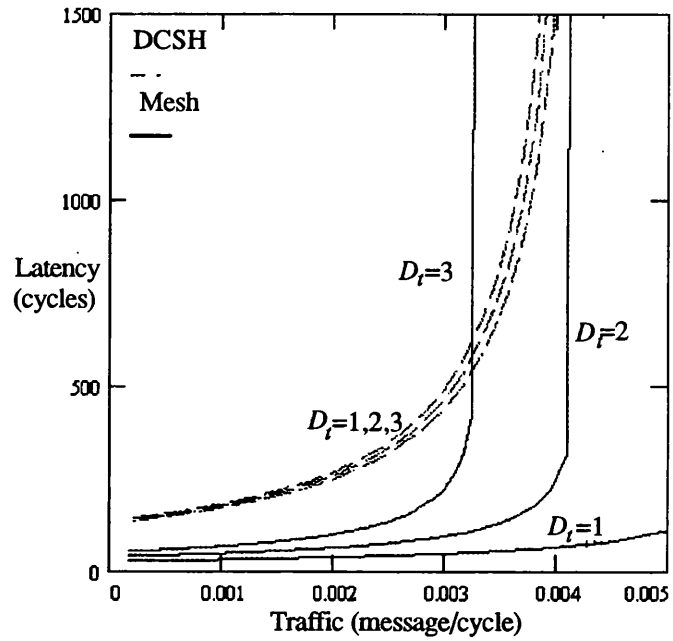



**Figure 5.8: Wormhole routing performance in the mesh & DCSH for a fixed message length ($M$=128 phits), $N$=256.**

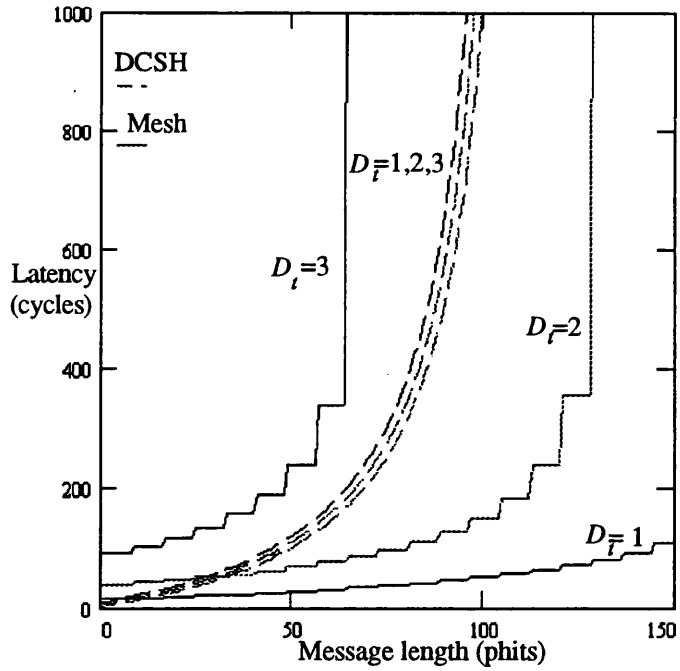



**Figure 5.9: Wormhole routing performance in the mesh & DCSH for a fixed generation rate ($m$=0.003 message/cycle), $N$=256.**

## Pin-out

The pin-out for the mesh is given by

$$P_{Mesh} = 4nW_{Mesh} \tag{5.29}$$

Similarly, for the DCSH it can be written as

$$P_{DCSH} = nkW_{DCSH} \tag{5.30}$$

To preserve the same pin-out in both networks, the channel width in the mesh should be

$$W_{Mesh} = \left\lceil \frac{k}{4} W_{DCSH} \right\rceil \tag{5.31}$$

## Virtual cut-through performance

Figures 5.10-12 reveal that the same conclusions are reached as under the previous constraint, even though the channel width ratio is reduced by half under this constraint.

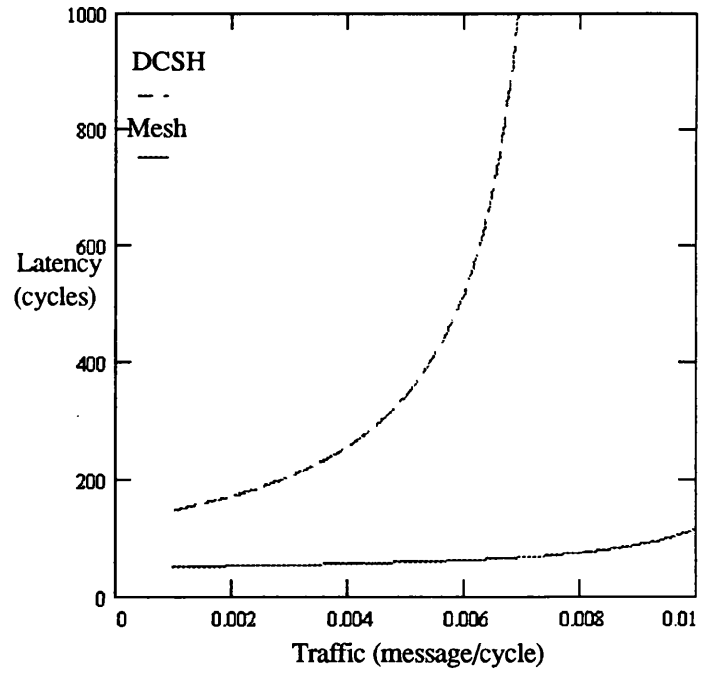



**Figure 5.10: Virtual cut-through performance in the mesh & DCSH for a fixed message length ($M$=128 phits).**

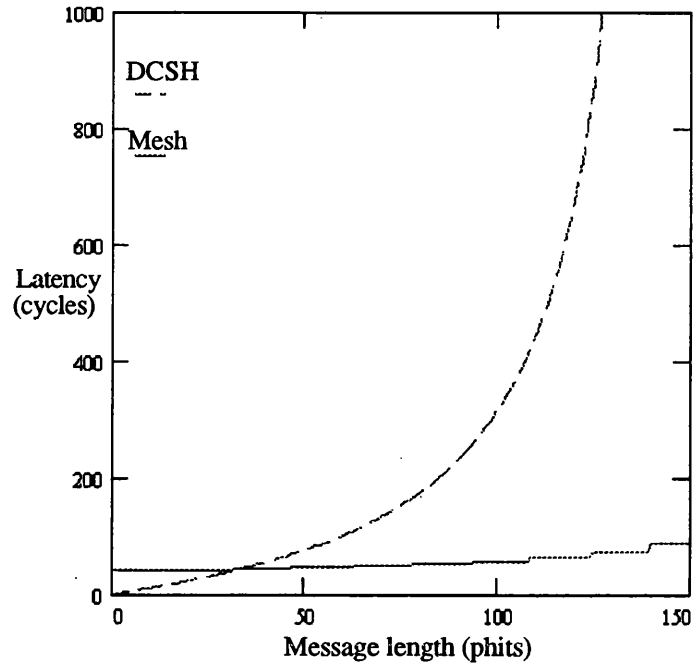



**Figure 5.11: Virtual cut-through performance in the mesh & DCSH for a fixed generation rate ($m$=0.005 message/cycle).**

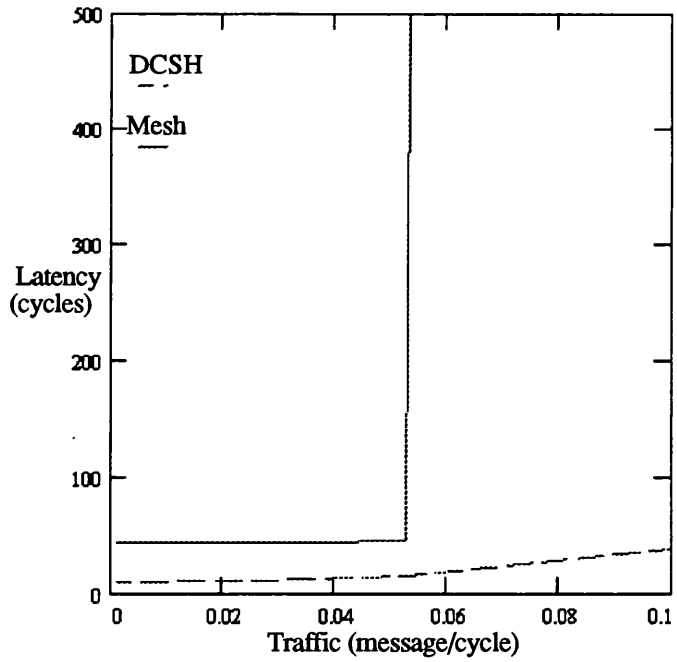



**Figure 5.12: Virtual cut-through performance in the mesh & DCSH for a fixed message length ($M$=8 phits).**

## Wormhole routing performance

Figures 5.13-16 show that for WR the same conclusions as before are reached, except that they are not so favourable to the mesh. For example, unlike under constant wiring

density, when the decision time is only 2 cycles the DCSH manages to outperform its mesh counterpart even for a small system size of 256 nodes.

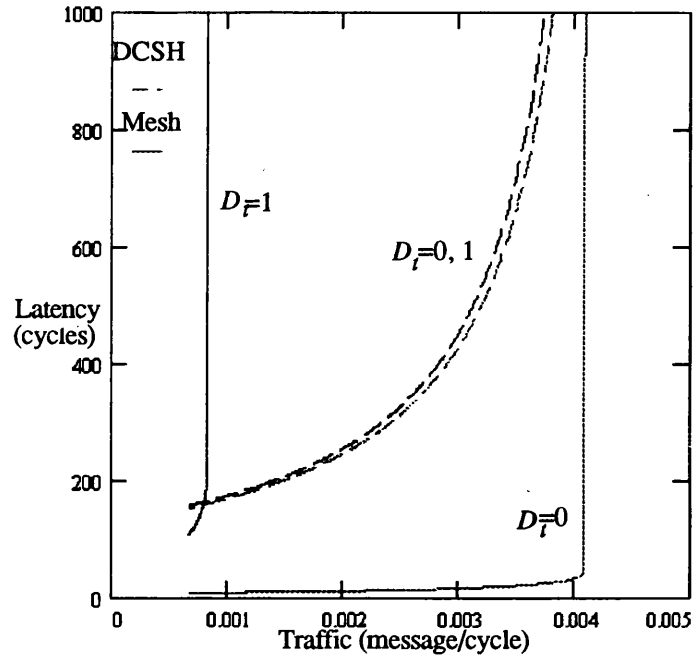



**Figure 5.13: Wormhole routing performance in the mesh & DCSH for a fixed message length ($M$=128 phits), $N$=4096.**

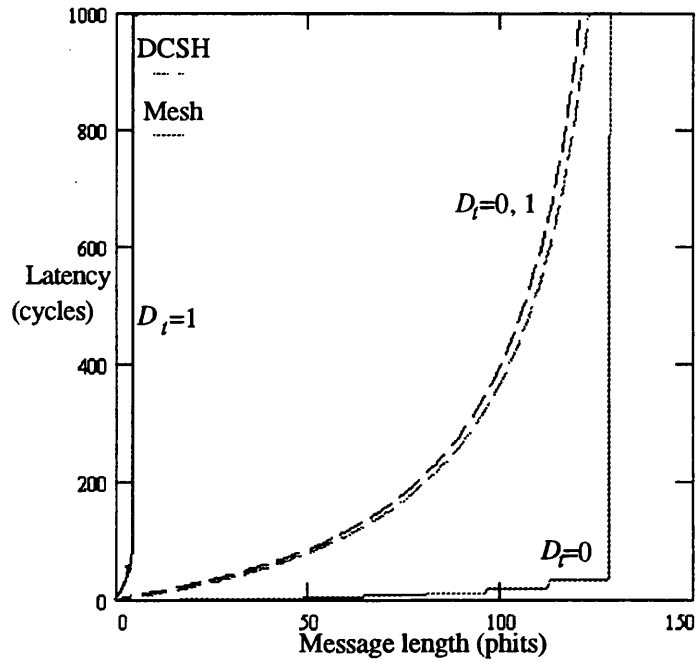



**Figure 5.14: Wormhole routing performance in the mesh & DCSH for a fixed generation rate ($m$=0.003 message/cycle), $N$=4096.**

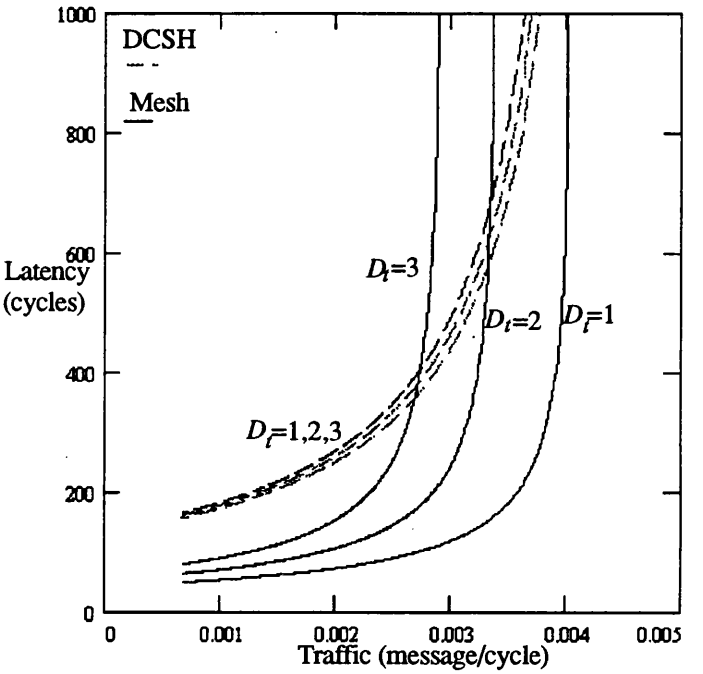



**Figure 5.15: Wormhole routing performance in the mesh & DCSH for a fixed message length ($M$=128 phits), $N$=256.**

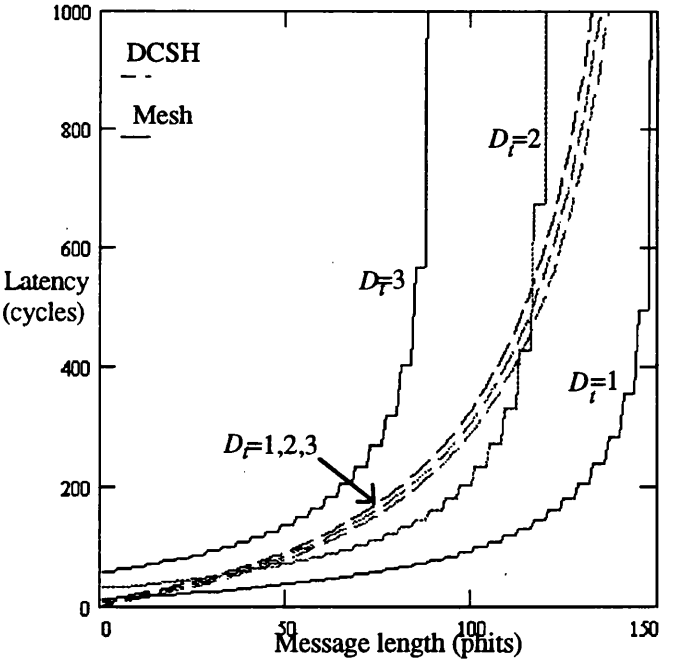



**Figure 5.16: Wormhole routing performance in the mesh & DCSH for a fixed generation rate ($m$=0.003 message/cycle), $N$=256.**

## 5.4 Merits of concurrent transmission in split-DCSHs

With VCT, the DCSH outperforms the mesh under the two given constraints when the message length is short. The mesh retains an advantage for longer messages. This section addresses the possibility of allowing a node in the DCSH to have concurrent message transmission in the two opposite directions of each cluster. The resulting structure, introduced in chapter 2 as the AS-DCSH, has an important advantage over the DCSH in that it has better utilisation of network channels while it retains the same wiring density. More importantly, the "switching concurrency" (i.e. the number of messages that can be simultaneously switched between dimensions) of an SE in the DCSH is increased to match that of the mesh.

### Wiring density

As the results depicted in Figures 5.17 and 5.18 show, the mesh still outperforms the AS-DCSH at long message lengths even though in the latter the traffic rate on channels has been reduced by half compared to its DCSH equivalent. However, this reduction in traffic rate is not sufficient to allow the message waiting time at an output queue in the AS-DCSH to be smaller than that in its mesh counterpart.

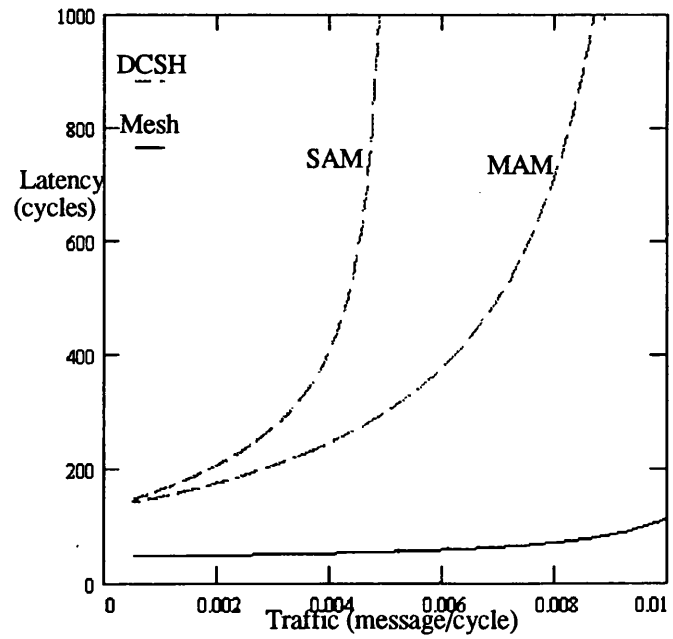



**Figure 5.17: Virtual cut-through performance in the mesh & AS-DCSH for a fixed message length ($M$=128 phits)[8].**

[8] There are one trace for the mesh because MAM and SAM yield the same results.

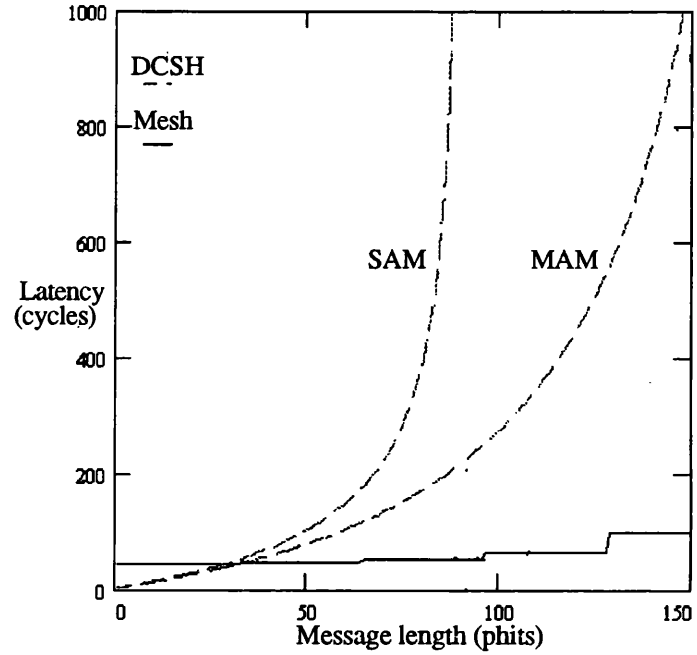



**Figure 5.18: Virtual cut-through performance in the mesh & AS-DCSH for a fixed generation rate ($m$=0.005 message/cycle)[8].**

## Pin-out

With MAM, the AS-DCSH outperforms the mesh under high traffic, as shown in Figure 5.19. This is because the traffic rate on its network channels is lower. The mesh can take advantage of its wider channel when the traffic is light and moderate. Figure 5.19 also shows that with SAM, the mesh has a lower latency at all traffic conditions because the larger message ratio of the DCSH makes the bottleneck which resides between the network and the local PE more severe.

With MAM, Figure 5.20 shows that when the message length is short causing the traffic to be light, the AS-DCSH outperforms the mesh. This is because the mesh cannot use its channel bandwidth efficiently when the message length is smaller than the channel width. As the message length increases slightly causing the traffic to become moderate, the mesh can use its wide channels effectively to reduce its latency. At long message lengths, the traffic becomes heavy and the AS-DCSH copes better with such a traffic condition. With SAM, however, the AS-DCSH is superior only at short messages.

The mesh is likely to be implemented using multiple-chip technology for the foreseeable future, and thus the constant pin-out constraint seems to be a more relevant constraint in practice than constant wiring density [23, 34, 55]. Further, this section shows that with

MAM, the AS-DCSH can have superior performance characteristics even at longer message lengths. The AS-DCSH does not requires a particularly complex SE design; requiring significantly fewer parallel local channels than, for example, the high-degree binary $n$-cube and generalised hypercube.

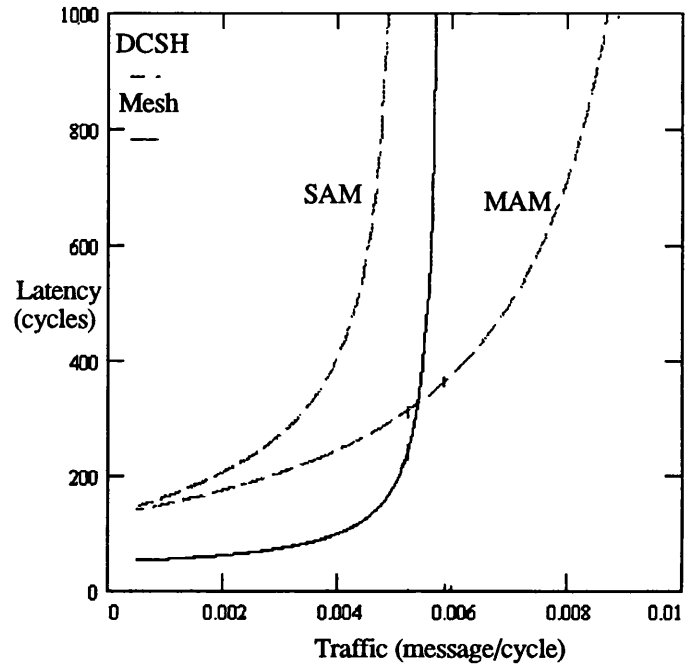



**Figure 5.19: Virtual cut-through performance in the mesh & AS-DCSH for a fixed message length ($M$=128 phits)[8].**

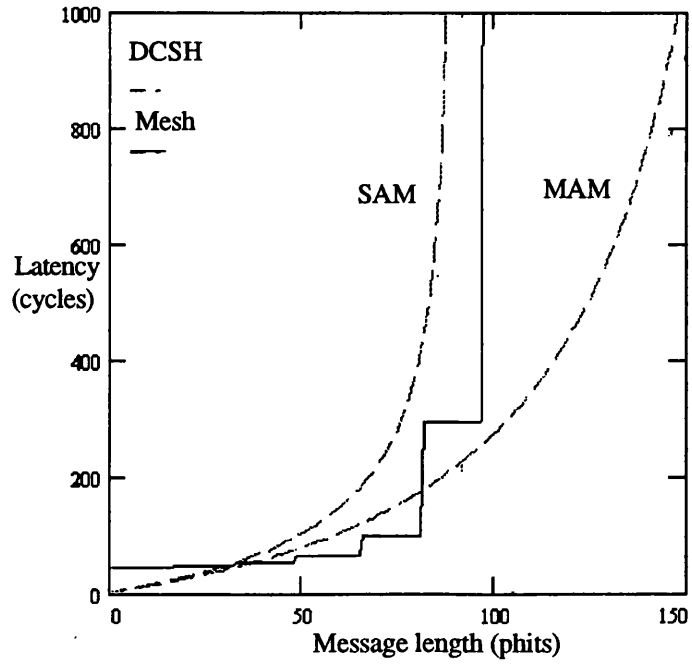



**Figure 5.20: Virtual cut-through performance in the mesh & AS-DCSH for a fixed generation rate ($m$=0.005 message/cycle)[8].**

## 5.5 Merits of the layered implementation

The DCSH can effectively escape channel width limitations by using the layered implementation scheme, although this is achieved at the expense of using more chips. The scheme cannot be fully exploited by the mesh due to limitations imposed by the relay bandwidth [56]. This section investigates to what extent the DCSH needs to exploit the layered implementation in order to have superior performance characteristics at long message lengths.

### Virtual cut-through performance

The DCSH need not increase its channel width to match that in the mesh in order to provide better performance. Figures 5.21 and 5.22 show that when the ratio between the channel width in the DCSH and mesh is 1:9 and 1:10, the mesh can still deliver a lower latency. However, when the ratio is 1:8 (*c.f.*, 1:32 under wiring density constraint and 1:16 under pin-out constraint) the DCSH outperforms the mesh.

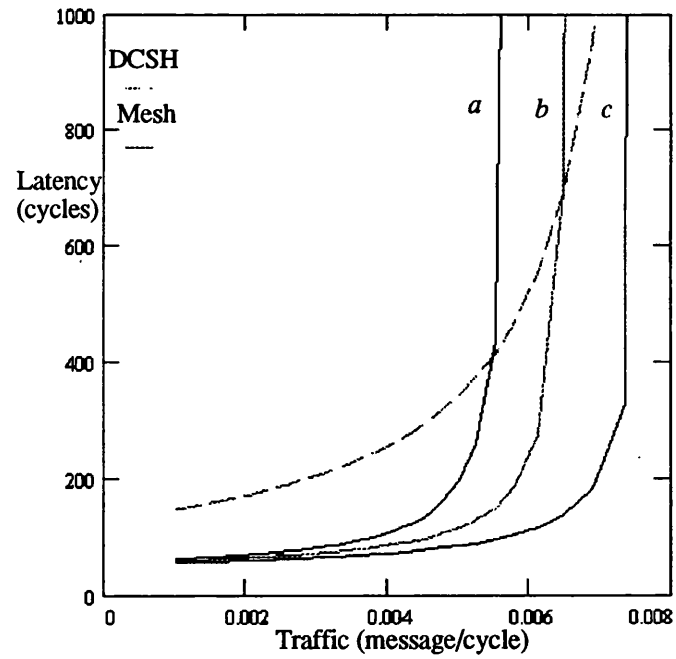



**Figure 5.21: Virtual cut-through performance in the mesh & DCSH for a fixed message length ($M$=128 phits), *a*) $W_{DCSH}$:$W_{SBH}$=1:8, *b*) $W_{DCSH}$:$W_{SBH}$=1:9, *c*) $W_{DCSH}$:$W_{SBH}$=1:10.**

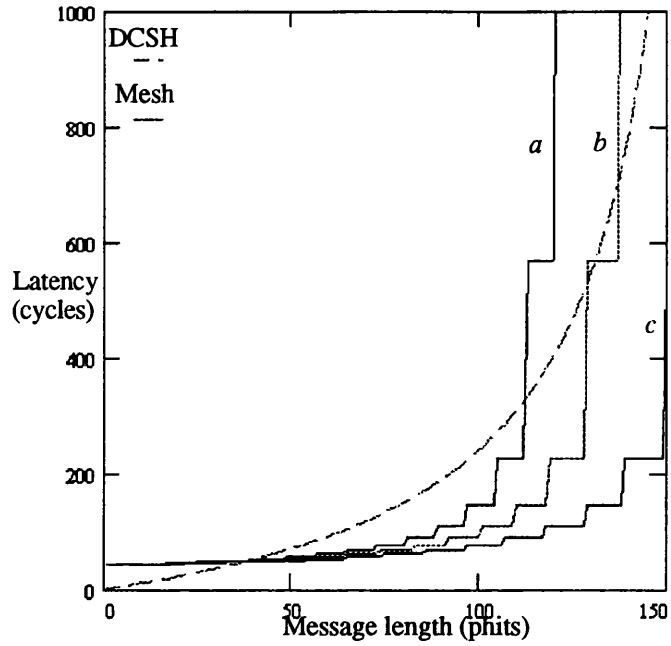



**Figure 5.22: Virtual cut-through performance in the mesh & DCSH for a fixed generation rate ($m$=0.005 message/cycle). *a*) $W_{DCSH}$:$W_{SBH}$=1:8, *b*) $W_{DCSH}$:$W_{SBH}$=1:9, *c*) $W_{DCSH}$:$W_{SBH}$=1:10.**

## Wormhole routing performance

When the channel width ratio is reduced to 1:13, which is very close to that under constant pin-out constraint, the DCSH outperforms the mesh under high traffic, even when the decision time is ignored (Figures 5.23 and 5.24). Further, when the ratio is further reduced to 1:12, the DCSH is better even at moderate traffic.

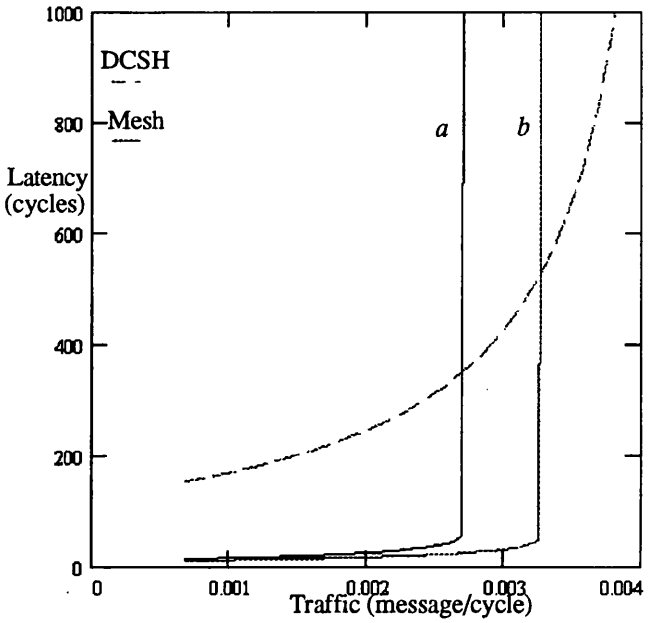



**Figure 5.23: Wormhole routing performance in the mesh & DCSH for a fixed message length ($M$=128 phits), *a*) $W_{DCSH}$:$W_{SBH}$=1:12, *b*) $W_{DCSH}$:$W_{SBH}$=1:13.**

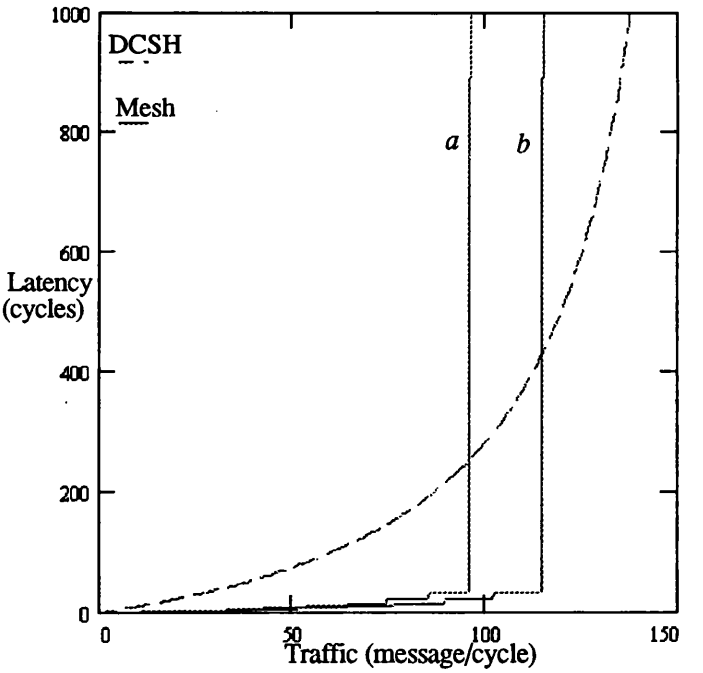



**Figure 5.24: Wormhole routing performance in the mesh & DCSH for a fixed generation rate ($m$=0.003 message/cycle), *a*) $W_{DCSH}$:$W_{SBH}$=1:12, *b*) $W_{DCSH}$:$W_{SBH}$=1:13.**

## 5.6 Conclusions

This chapter has assessed the relative merits of the low-dimensional mesh and its DCSH counterpart. The former is suitable for applications that exhibit communication locality. However, modifications to the basic structure have been proposed to reduce the deleterious effects of its high diameter in applications that require non-local communication (e.g., broadcast operations). Further, layered mesh implementations are limited by the high relay bandwidth requirements which prevent the efficient use of the increased channel width. DCSHs, in contrast, are inherently suitable for both nearest-neighbour communications and broadcast/non-local communication. Moreover, the relay bandwidth constraint in DCSH networks is relaxed due to the multiplexing-down effects inherent in their topologies.

With virtual cut-through, whether VLSI or multiple chip technology is used, meshes have a lower latency than their DCSH counterparts only for long messages. DCSHs deliver superior performance when high wiring densities and pin-outs are available, permitting low message aspect ratios. Further, as the network size scales, the advantages of DCSHs at short messages becomes even more pronounced.

Most current commercial multicomputers are based on meshes which use wormhole routing. The previous sections have shown that under constant wiring and pin-out constraints, DCSHs have superior performance characteristics to meshes at all message lengths. Meshes, particularly large-scale ones, are extremely sensitive to decision time. With realistic estimates of decision time, the performance of meshes degrades considerably due to the number of switching elements that a message visits. DCSHs, in contrast, are almost totally insensitive to the effects of decision time. DCSHs, therefore, can use slower and cheaper switching elements, and still deliver better performance than meshes.

Finally and most significantly of all, the layered implementation scheme enables DCSHs to escape the constraints imposed by wiring density and pin-out, and therefore can increase their channel width. This technique effectively allows DCSHs to deliver better performance than meshes using any switching technique and any message length.

# 6 Performance comparison with other important networks

## 6.1 Introduction

Several interconnection networks have been proposed in the past two decades and have been used in real machine designs. These include are the multi-stage network (MIN) and binary $n$-cube (cube for short). The former has often been suggested for shared-memory multiprocessors such as the Cedar [139], BNN Butterfly [140], RP3 [141], and Ultracomputer [142]. The latter has been very popular in first generation multicomputers such as the Cosmic Cube [42], iPSC [40], and Ametek III [143].

This chapter compares the performance of the MIN to the DCSH, assuming equal implementation cost using VLSI and multiple-chip technology. Chapter 2 has shown that unlike their higher-dimensional counterparts, low-dimensional DCSHs can exploit a new implementation to increase their channel width. This chapter, however, assesses whether they can provide superior performance to their higher-dimensional counterparts when using more conventional implementation schemes.

## 6.2 Multi-stage interconnection networks (MINs)

The MIN connecting $N$ ($=k^n$) nodes is made up of $n$ stages of $k \times k$ crossbar switches.

Every message has to travel all $n$ stages to reach its destination (Figure 6.1). The processing elements (PEs) are situated only at the first and last stage. The MIN provides uniform-cost access to global shared memory modules in shared-memory multiprocessors. However, they do not allow exploitation of the communication locality which has been found in several parallel applications [3, 119, 120].

In the past, the MIN has been analysed extensively in the literature and several variations have been proposed [84, 85, 86]. It has been shown that some of these variations are equivalent in a SIMD environment [86, 144]. Hypermeshes, including the DCSH, can also be shown to be equivalent to some MINs, such as the Omega and inverse Omega [87, 102].

Although Knight *et al* have devised a layered implementation scheme for the MIN [141], the DCSH provides a more general structure due to its ability to support both local and non-local communication.

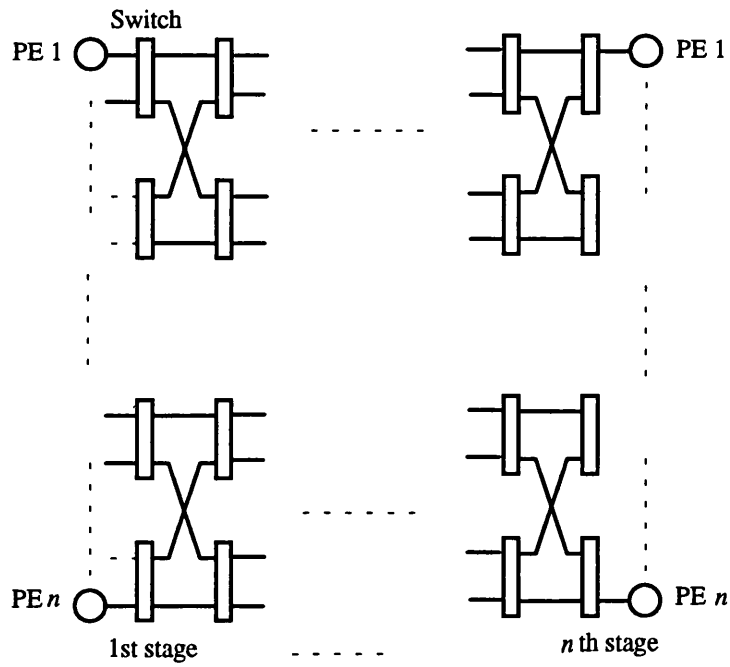



**Figure 6.1: An example of a MIN using 2×2 switches.**

### 6.2.1 Mathematical models for MINs

A node generates a fixed length ($B$ flits) message with probability $m$ in a cycle. Messages are destined to the other ($N$-1) nodes with equal probability.

#### Virtual-cut through model

Kruskal and Snir have derived a queueing model for the MIN to predict the latency in a message-switching environment [122]. The modelcan easily be adapted for VCT. The mean latency to cross an $n$-stage MIN can be written as

$$L = n(D_t + w_o) + B \tag{6.1}$$

where $D_t$ is the decision time and $w_o$ is the mean waiting time at an output queue

$$w_o = \frac{\rho}{2(1-\rho)}(1-\frac{1}{k})B \tag{6.2}$$

$\rho$ is the channel utilisation

$$\rho = mB\frac{N}{(N-1)} \tag{6.3}$$

(the factor $N/(N\text{-}1)$ has been included here because nodes do not reference themselves).

#### Wormhole routing model

The model developed by Dally is used here with some modifications [38]. Since nodes do not reference themselves, the channel traffic rate is

$$m' = \frac{N}{(N-1)}m \tag{6.4}$$

Let messages visit the $n$ stages in decreasing order, with the destination node at stage 0. Since flits at the destination are serviced as soon as they arrive, the latency is given by

$$L_0 = B \tag{6.5}$$

Since phits of a single message may be spread across several stages, the latency $L_i$ seen by a message when crossing a switch at stage $i$ is given in terms of the latency at the following stage. Equations 3.31 (Section 3.4.2) gives $L_i$ as

$$L_{i+1} = L_i + \frac{(k-1)m'}{2k} L_i^2 + D_t \tag{6.6}$$

## 6.2.2 Performance comparison of MINs and DCSHs

Hereafter, performance results are presented for $N$=4096 nodes. The discussion focuses on a low-dimensional case, $n$=2 (although brief conclusions are also presented for higher-dimensional topologies if they are different). For VCT, only the multiple-accepting model (MAM) is used in the DCSH (results for SAM are shown if they are different). The decision time is ignored as it does not greatly change the general conclusions.

### Wiring density

The bisection width of the MIN with a channel width of $W_{MIN}$ is given by [38]

$$W_{MIN} = NW_{MIN} \tag{6.7}$$

since $k$ channels are cut corresponding to each $k \times k$ switch when slicing the MIN into equal parts, and the total number of such cuts is therefore $N$. Section 4.2.2 has shown that the bisection width of the DCSH with a channel width of $W_{DCSH}$ is

$$W_{DCSH} = NW_{DCSH} \tag{6.8}$$

With equal implementation cost in VLSI, the channel width in the MIN is found to be

$$W_{MIN} = W_{DCSH} \tag{6.9}$$

Varying the dimensionality of the DCSH and the number of stages in the MIN, while keeping the same network size, gives different topologies. However, the channel width in the DCSH and MIN is the same.

### Virtual cut-through performance

Figure 6.2 shows the mean latency as a function of the generation rate with fixed message length in the 2-stage MIN (2S-MIN for short) and the 2-D DCSH. Both networks have a comparable latency under light and high traffic. This is because both networks have almost identical channel traffic rate and average message distance when the dimensionality is low. Under moderate traffic, however, the 2-S MIN has a slightly lower latency because more message queueing occurs in the 2D- DCSH.

Figure 6.3, which shows the effects of message length on the system performance when the generation rate is kept constant, reveals that the 2S-MIN and 2-D DCSH have almost the same performance at different message lengths.

(It is worth noting that when the dimensionality is increased to $n$=6, the 6-D DCSH outperforms the 6-S MIN. This is due to the fact that the average distance and the channel traffic rate are lower in the former. The largest difference in performance in the favour of the DCSH occurs for the highest dimension, $n$=12 for $N$=4096.)

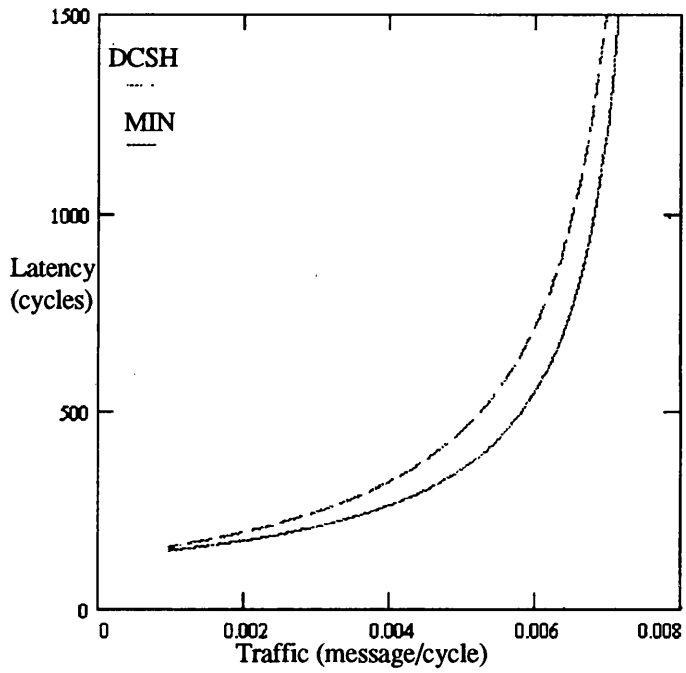



**Figure 6.2: Virtual-cut through performance in the MIN & DCSH for a fixed message length ($M$=128 phits).**

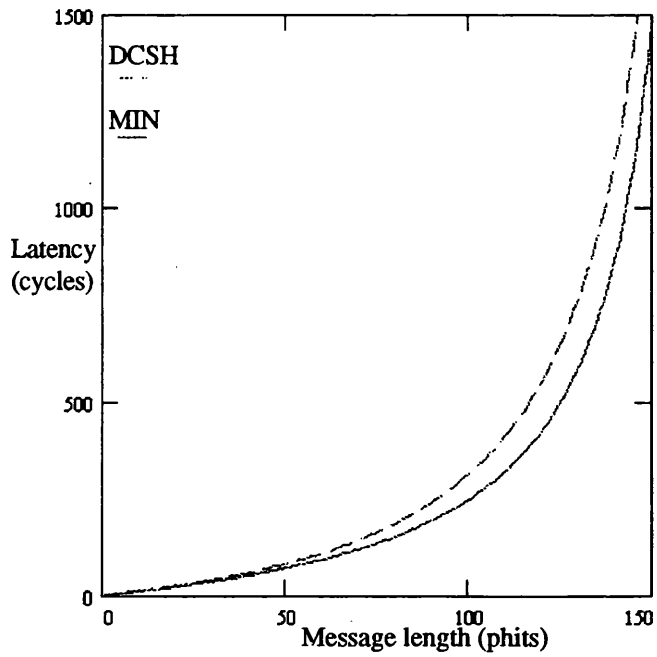



**Figure 6.3: Virtual-cut through performance in the MIN & DCSH for a fixed generation rate ($m$=0.006 message/cycle).**

## Wormhole routing performance

The 2-D DCSH and 2-S MIN have the same performance (Figure 6.4 and 6.5). Section 3.4 has shown that a message in a DCSH encounters two blocking stages at an SE. The first stage occurs when a message, arriving at a multiplexer, has to compete to be selected by the multiplexer. The second stage occurs when a message has to compete for a network channel when crossing a dimension. The latency seen by a message at these blocking stages is given by Equations 3.12 and 3.16 respectively. A message in a MIN goes through one blocking stage in a switch and Equation 6.2 gives the corresponding latency. Equation 3.12 reveals that when $k$ is large (as in the 2-D DCSH) message blocking when crossing a dimension becomes negligible, and consequently, only the blocking which occurs at the multiplexer affects the latency. Thus, the number of blocking stages in both the 2-D DCSH and 2-S MIN is the same. Further, a message encounters the same blocking in either network.

(When the dimensionality is increased to 6, the 6-D DCSH has a lower latency than the 6-S MIN. This is because blocking encountered at a multiplexer and channel in the 6-D DCSH is lower than that encountered at a switch in the 6-S MIN.)

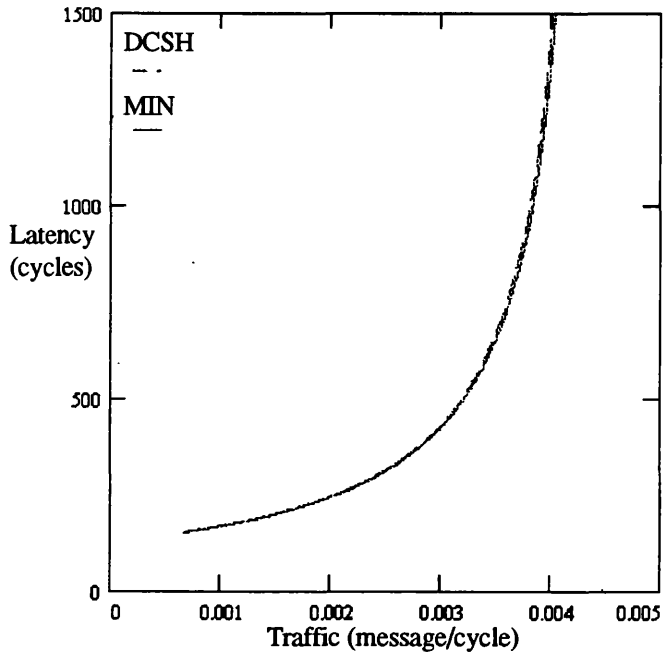



**Figure 6.4. Wormhole routing performance in the MIN & MIN for a message length ($M$=128 phits).**

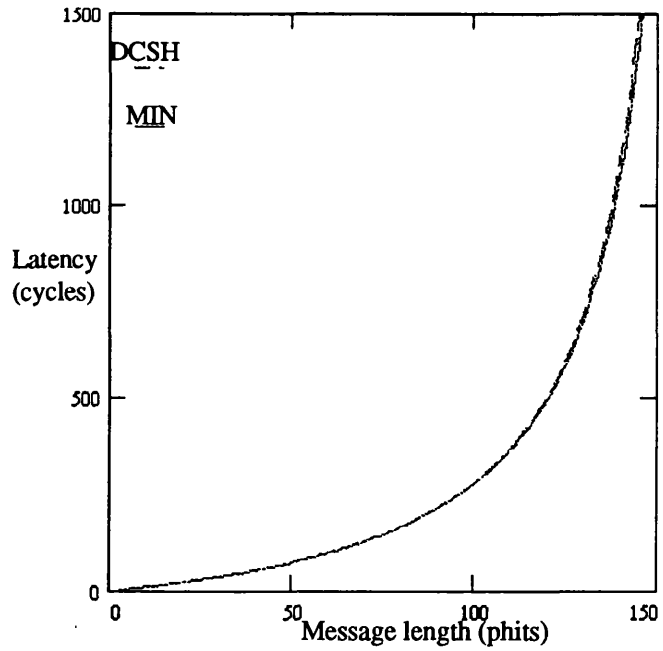



**Figure 6.5. Wormhole routing performance in the MIN & DCSH for a fixed generation rate ($m$=0.003 message/cycle).**

### Pin-out

The pin-out of a $k \times k$ switch is the primary cost measure when the MIN is implemented over multiple chips. Assuming that a switch is implemented on a single chip, the switch pin-out can be written as

$$P_{MIN} = 2kW_{MIN} \tag{6.10}$$

The DCSH has a pin-out of

$$P_{DCSH} = nkW_{DCSH} \tag{6.11}$$

To preserve the same pin-out in both networks, the following should be satisfied

$$W_{MIN} = \left\lceil \frac{n}{2} W_{DCSH} \right\rceil \tag{6.12}$$

The 2-S MIN and 2-D DCSH then again have comparable performance since they have the same channel width, as under the constant wiring density constraint. (When $n$=6, the 6-S MIN outperforms the 6-D DCSH because of its wider channels.)

## 6.2.3 Communication locality

Unlike the MIN, the DCSH can take advantage of the communication locality which occurs in several parallel applications. Section 3.6 has shown that communication locality improves the DCSH performance, since local messages consume less network bandwidth because of their short travel distance. The following sections present a comparison of the DCSH and MIN under traffic with locality. The "sphere of locality" model is used to model communication locality [3, 119].

### Wiring density

It will be assumed that the size of the sphere of locality $c$=1 and the fraction of the traffic within the sphere is $\beta = 50\%$. The results are shown for MAM, since Section 3.6 has shown that SAM offsets any benefits gained from exploiting locality at high traffic.

### Virtual-cut through performance

The 2-D DCSH outperforms the 2-S MIN (Figures 6.6 and 6.7). Local traffic travels a shorter distance and therefore encounters less blocking. Further, as non-local traffic is reduced, so is message blocking. These factors together reduce delays in the 2-D DCSH.

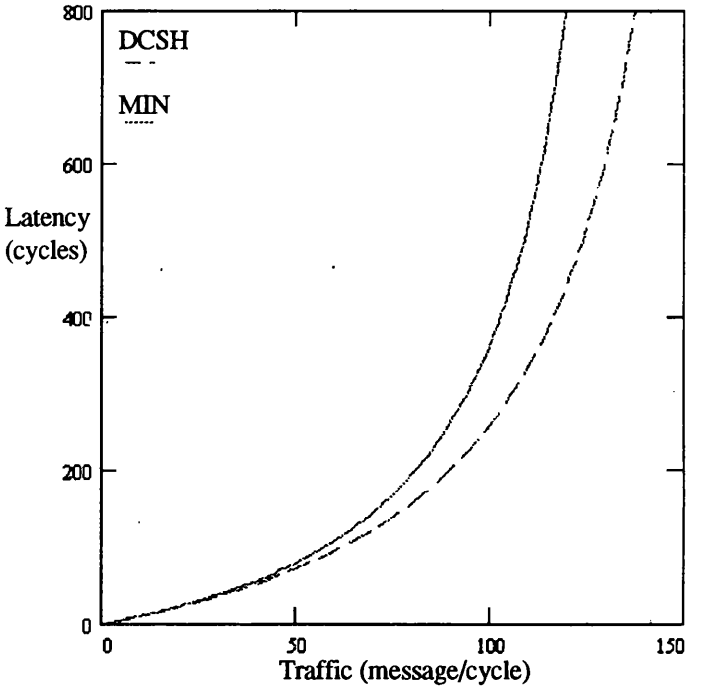



**Figure 6.6: Virtual cut-through performance with locality in the MIN & DCSH for a fixed message length ($M$=128 phits), $c$=1, β=50%.**

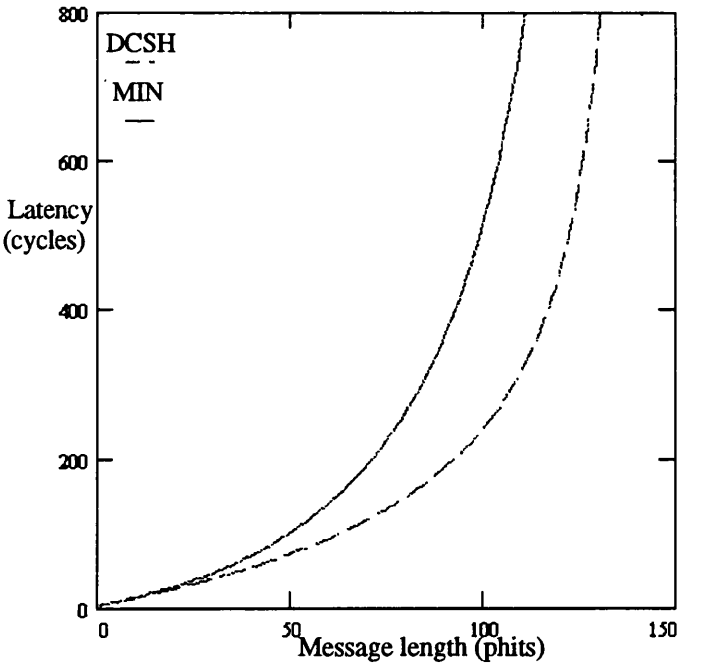



**Figure 6.7: Virtual cut-through performance with locality in the MIN & DCSH for a fixed generation rate ($m$=0.006 message/cycle), $c$=1, β=50%.**

## Wormhole routing performance

As with VCT, the 2-D DCSH outperforms the 2-S MIN since communication locality reduces message blocking (Figures 6.8 and 6.9).

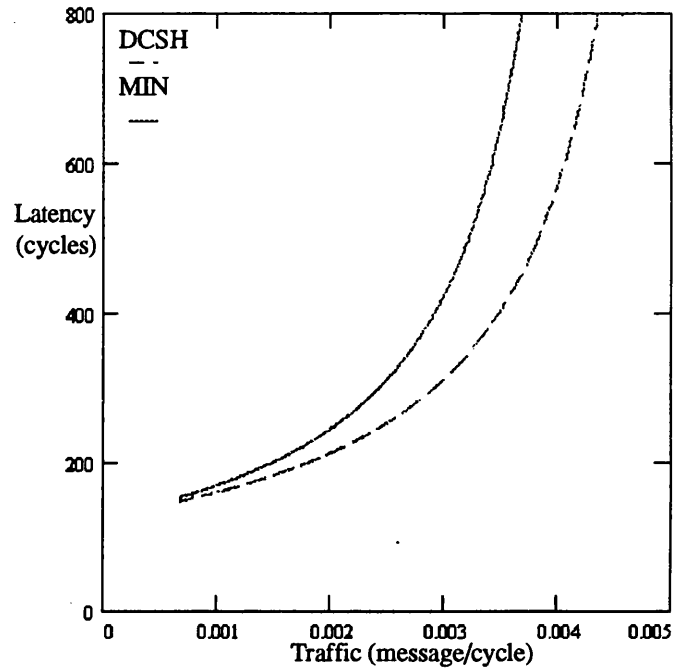



**Figure 6.8: Wormhole routing performance with locality in the MIN & DCSH for a fixed message length ($M$=128 phits), $c$=1, β=50%.**

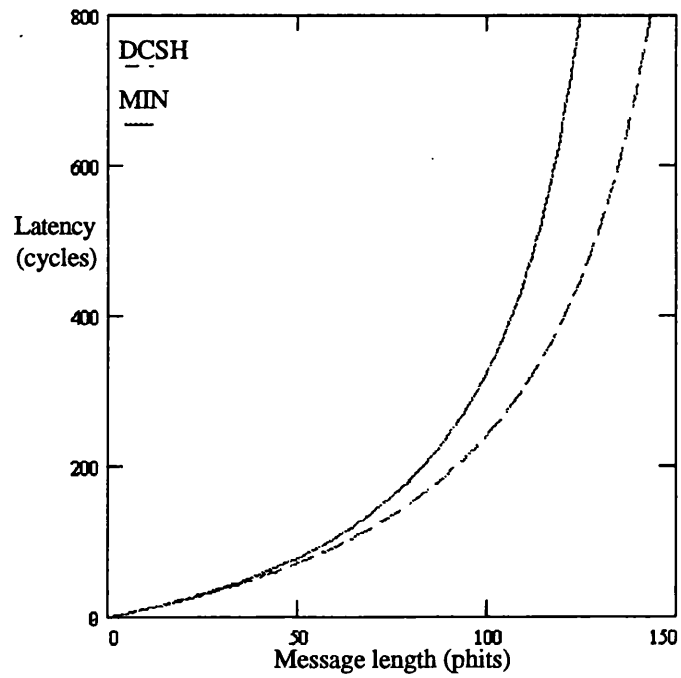



**Figure 6.9: Wormhole routing performance with locality in the MIN & DCSH for a fixed generation rate ($m$=0.003 message/cycle), $c$=1, β=50%.**

### Pin-out

The results for the 2-dimensional case is similar to those presented under the wiring density constraint. (Results for higher-dimensional DCSHs have revealed that they cannot outperform their MIN counterparts because of their thinner channel width.)

## 6.3 Binary *n*-cubes

As mentioned in Chapter 2, the binary *n*-cube is a degenerate limiting case of the DCSH family, and therefore comparison between DCSHs and cubes may not be a logical undertaking. However, for a given number of nodes, *N*, cubes (hypermeshes of high dimensionality) can be compared with low-dimensional DCSHs. This is the final objective of the current chapter.

The small diameter of binary *n*-cubes minimises the buffering overhead associated with the message switching widely used in first generation multicomputers [40, 41, 42, 143]. As discussed before, these networks have lost ground to the torus and mesh more recently because Dally has shown that they perform less well when implemented in VLSI [22, 37, 38]. However, neither current and foreseeable technology can integrate a large number of nodes on a single VLSI-chip, and therefore Dally's study is as yet of little practical value. Abraham and Padmanavan have shown that when networks are laid out on several chips, binary *n*-cubes are superior to meshes and tori [23, 34, 49]. Nevertheless, several more experimental and commercial machines have still been based on the latter structures. The main reason behind this choice is that mesh and torus structures are more modular and easier to wire because of their low-dimensionality.

Low-dimensional DCSHs have several important advantages over their higher-dimensional counterparts, including binary *n*-cubes [Chapter 3]. Firstly, low-dimensional DCSHs can use random routing, which is fault-tolerant and has the same performance as restricted routing, while cubes must use restricted routing for optimal performance. Secondly, low-dimensional DCSHs can use the single-accepting model and still provide the same performance as with the multiple-accepting model. However, cubes must use the multiple-accepting model or the advantage of their rich connectivity is lost. This gives the low-dimensional DCSH a significant advantage over the cube in node costs. Thirdly, low-dimensional DCSHs can further reduce their implementation cost by using switching elements with input buffering only; these deliver comparable performance to SEs with both input and output buffering. Cubes, however, need both input and output

buffering to maintain good performance.

Practical systems based on the cube suffer from channel bandwidth limitations because of the large number of channels [40, 41, 42]. Further, the cube cannot exploit a layered implementation because of the difficulty in separating switching from processing due to the complex interconnection pattern. By exploiting a layered implementation low-dimensional DCSHs can have wider channels than the cube, and therefore will have superior performance. It would be pointless to labour this superiority further and more interesting comparisons between the topologies can be obtained by using the assumptions of more conventional implementations, as done in the following section.

### 6.3.1 Comparison of binary *n*-cubes and lower-dimensional DCSHs

The 2-dimensional DCSH is evaluated against the cube. Results are presented for a moderately large system with $N$=4096. For VCT, both MAM and SAM are considered.

#### Wiring density

Equation 6.8 shows that the DCSH has a fixed bisection width which is independent of the dimensionality. Thus, the 2-D DCSH and cube have the same channel width. (It is worth noting that the peak width in the cube is higher than the bisection width and is given by $\lceil 4/3N \rceil$ [98, 130], while in the 2-D DCSH it is the same as its bisection width. Therefore, the use of bisection width instead of peak width gives the cube a slight unfair advantage. Results using peak width are reported if they are different from those using bisection width)

#### Virtual-cut through performance

With MAM, the cube has a lower latency than the 2-D DCSH, as shown in Figure 6.10. This is because the traffic rate on cube channels is lower. With SAM, the cube has a lower latency under moderate traffic. Both networks, however, have comparable performance under high traffic. Most existing parallel systems based on cubes have avoided using MAM because it is expensive to implement. Using SAM, therefore, represents a more realistic practical situation, and reflects the fact that the rich connectivity of the cube is not cost-effective and cannot be fully exploited.

Figure 6.11 reveals that the cube has better performance at long message lengths.

However, at short message lengths the 2-D DCSH is superior, since the former cannot take the full benefit of its large number of channels. Figure 6.12 shows that the cube has no advantage over the 2-D DCSH when high wiring densities and pin-out counts are available because they favour short message aspect ratios.

Using the peak width of the cube can yield different results, as shown in Figure 6.13 and 6.14. With SAM, the 2-D DCSH is superior to the cube for all message lengths due to the larger message aspect ratio of the latter, which makes the PE-SE bottleneck worse. With MAM, on the other hand, the results do not change from those when the bisection width is used.

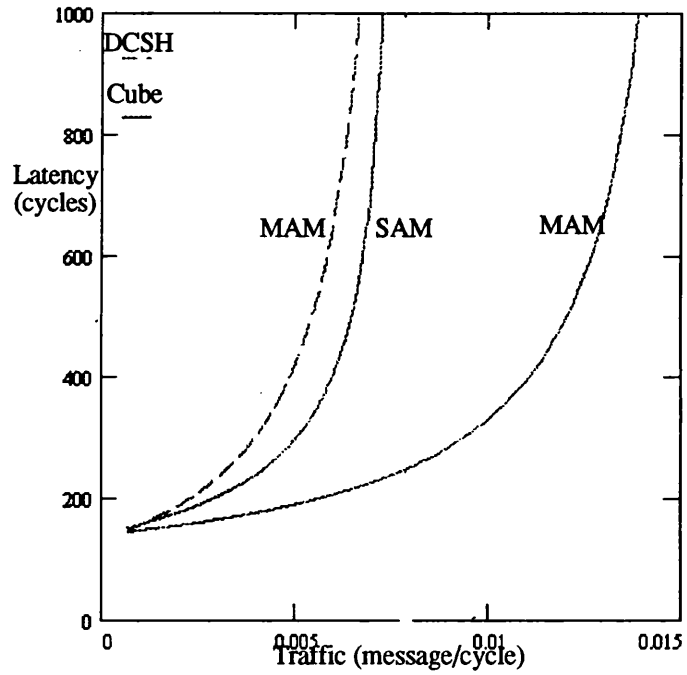



**Figure 6.10: Virtual cut-through performance in the cube & 2-D DCSH using bisection width and a fixed message length ($M$=128 phits)[7].**

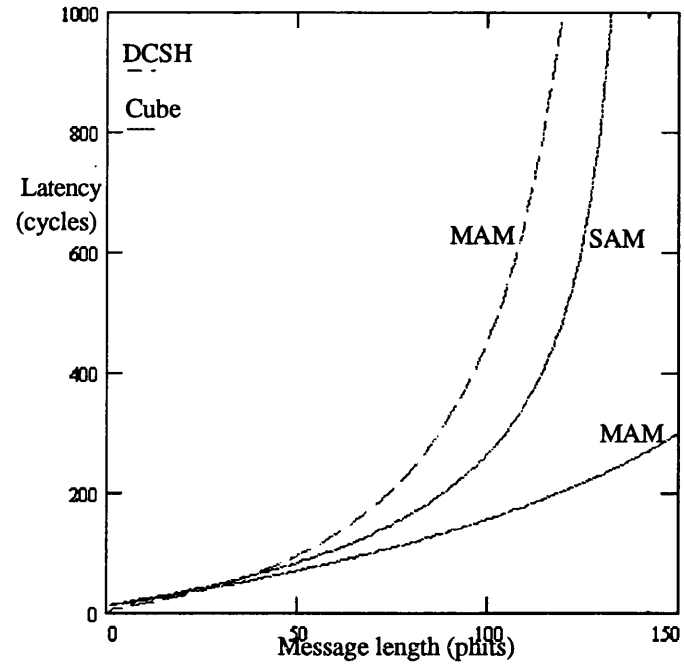



**Figure 6.11: Virtual cut-through performance in the cube & 2-D DCSH using bisection width and fixed generation rate ($m$=0.006 message/cycle)[7].**

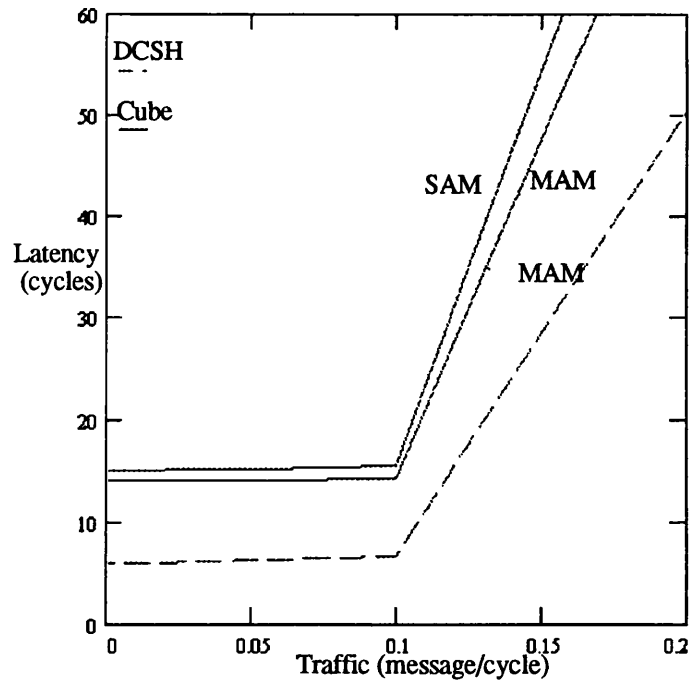



**Figure 6.12: Virtual cut-through performance in the cube & 2-D DCSH using bisection width and a fixed message length ($m$=4 phits)[7].**

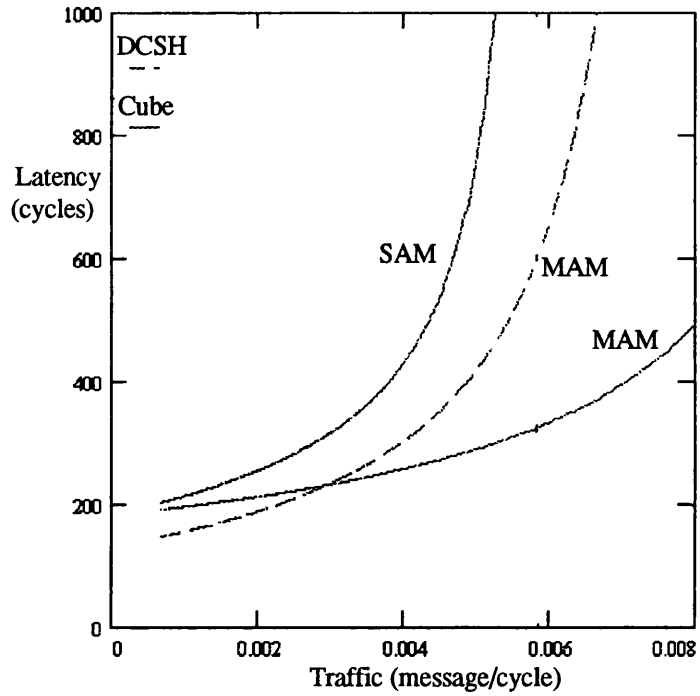



**Figure 6.13: Virtual cut-through performance in the cube & 2-D DCSH using peak width and a fixed message length ($m$=128 phits)[7].**

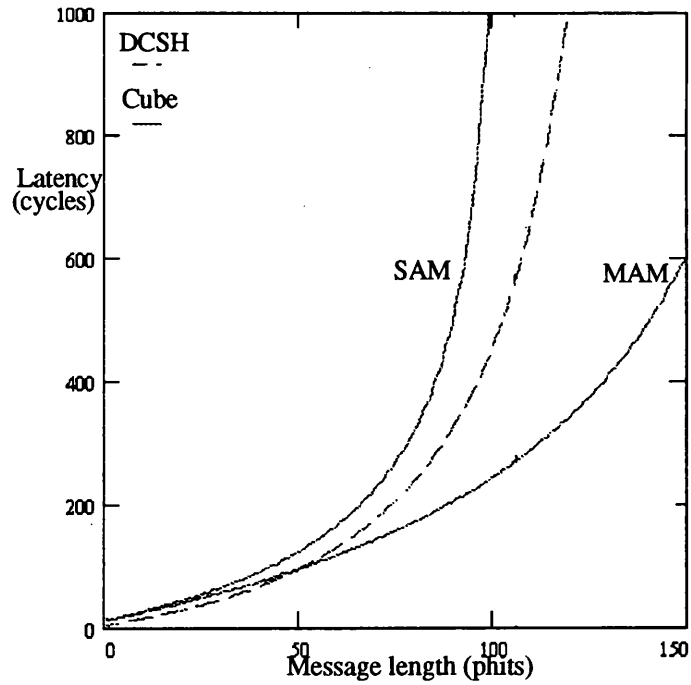



**Figure 6.14: Virtual cut-through performance in the cube & 2-D DCSH using peak width and fixed generation rate ($m$=0.006 message/cycle)[7].**

## Wormhole routing performance

The 2-D DCSH has a lower latency than the cube (Figure 6.15 and 6.16). This is because message blocking in the latter is higher because a message has to go through a larger number of switching elements.

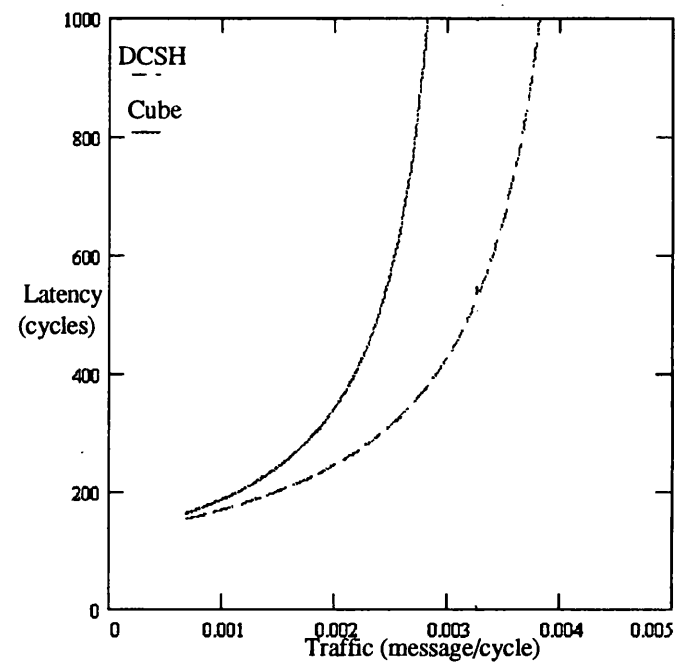



**Figure 6.15: Wormhole routing performance in the cube & 2-D DCSH for a fixed message length ($M$=128 phits).**

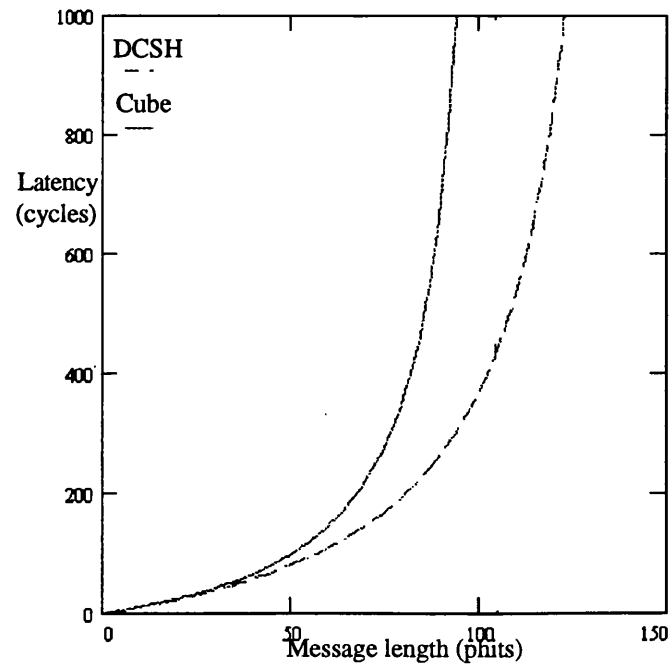



**Figure 6.16: Wormhole routing performance in the cube & 2-D DCSH for a fixed generation rate ($m$=0.004 message/cycle).**

## Pin-out

Equation 6.15 gives the pin-out in a $k^n$ node DCSH. To preserve the same pin-out between the 2-D DCSH and cube, the channel width in the cube should satisfy

$$W_{Cube} = \left\lceil \frac{k_2}{n_{12}} W_{2D\,DCSH} \right\rceil \tag{6.13}$$

where $2^{n_{12}} = k_2^2 = N$ ($n_{12}$ is the dimensionality of the cube and $k_2$ the width of the 2D DCSH).

## Virtual cut-through performance

The cube has better performance than the 2-D DCSH at long message lengths with both SAM and MAM because of its wider channels (Figures 6.17 and 6.18). However, the advantage of the cube diminishes at short message lengths (Figure 6.19).

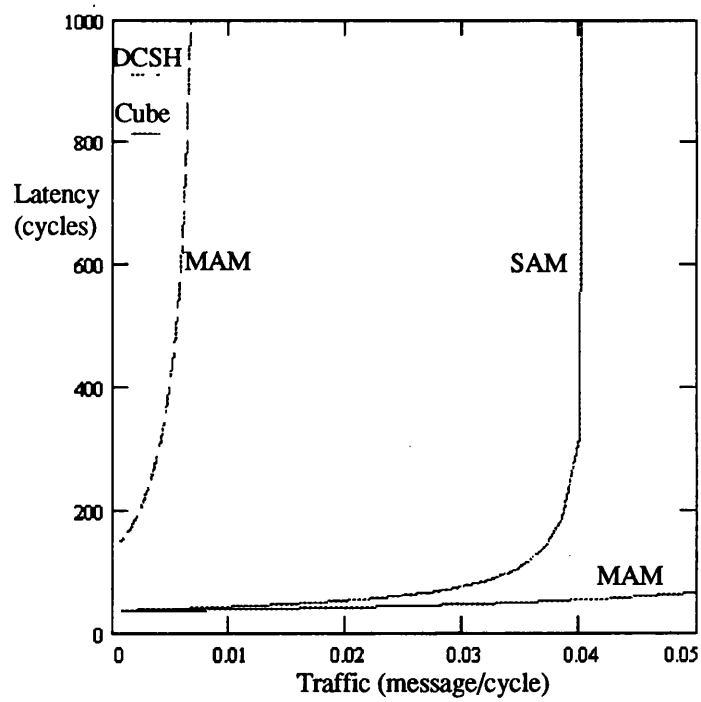



**Figure 6.17: Virtual cut-through performance in the cube & 2-D DCSH for a fixed message length ($M$=128 phits)[7].**

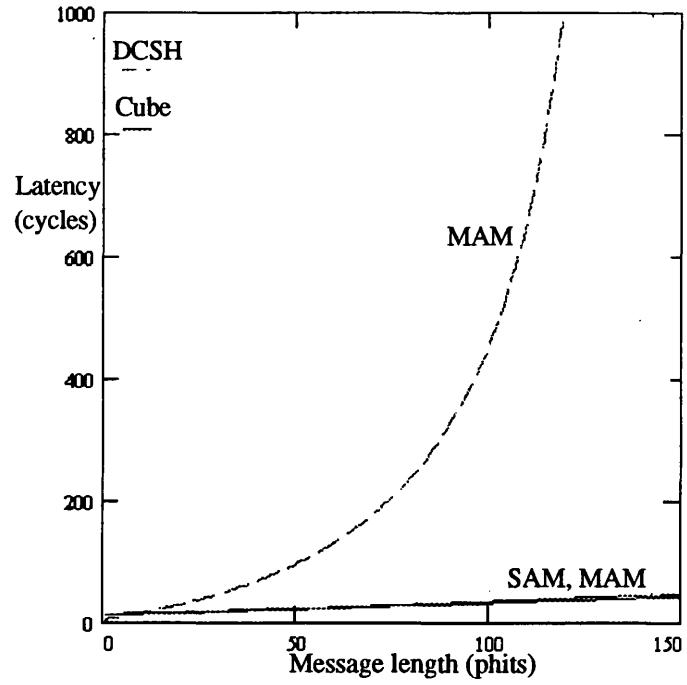



**Figure 6.18: Virtual cut-through performance in the cube & 2-D DCSH for a fixed generation rate ($m$=0.006 message/cycle)[7].**

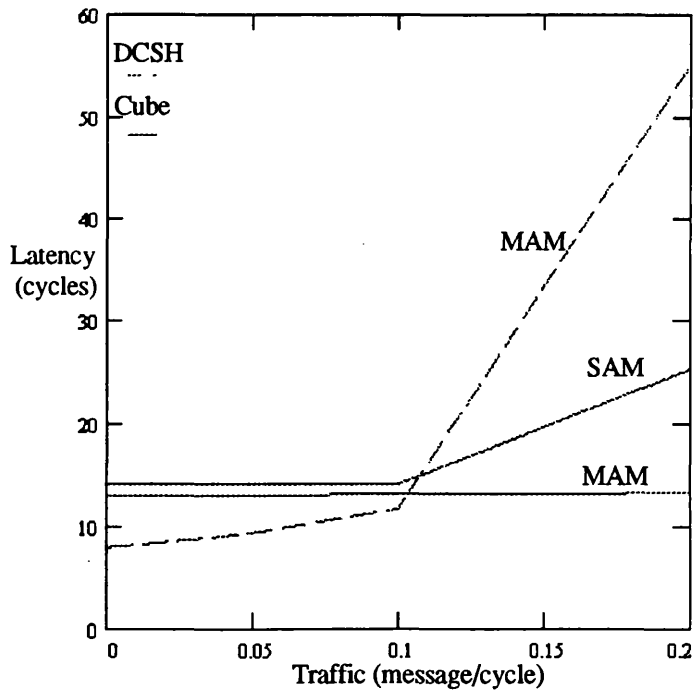



**Figure 6.19: Virtual cut-through performance in the cube & 2-D DCSH for a fixed message length ($M$=4 phits)[7].**

## Wormhole routing performance

The cube outperforms the 2-D DCSH at long message lengths, even when the decision time is taken into account, as shown in Figure 6.20 and 6.21. This is because the message aspect ratio in the 2-D DCSH is larger, resulting in higher message blocking.

When the message length decreases, however, the 2-D DCSH outperforms the cube when the decision time is realistic. Figure 6.22 for $M$=16 phit and Figure 5.23 for $M$=8 phits illustrate this with different decision times. The two figures also reveal that as the message length decreases, the deleterious effects of decision time on performance increase. This is because a short message reduces the ratio between the message aspect ratios in both networks. Further, the cube is more sensitive to decision time because a message goes through more SE and therefore its performance degrades considerably. These two factors together favour the 2-D DCSH at short messages.

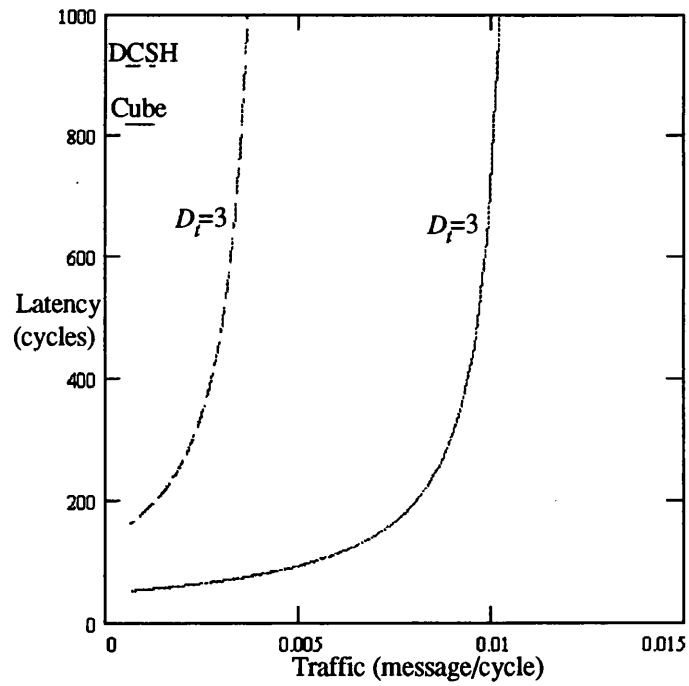



**Figure 6.20: Wormhole routing performance in the cube & 2-D DCSH for a fixed message length ($M$=128 phits).**

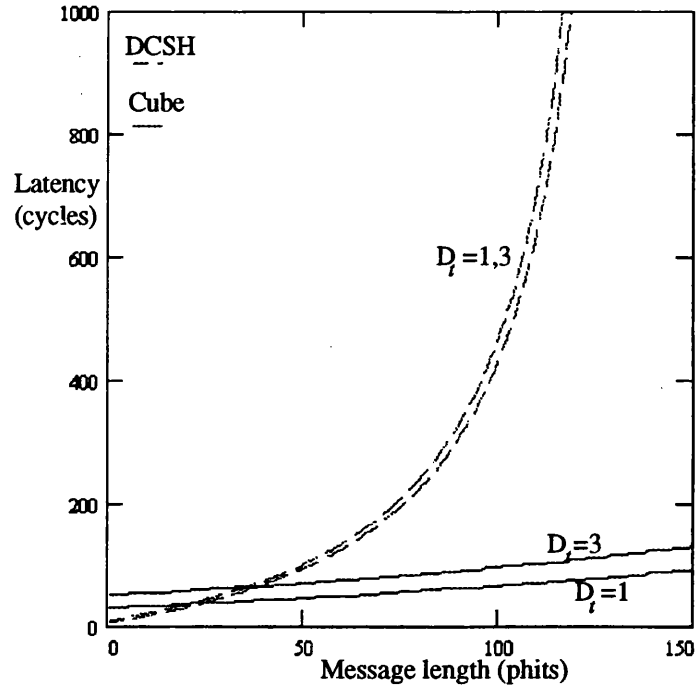



**Figure 6.21: Wormhole routing performance in the cube & 2-D DCSH for a fixed generation rate ($m$=0.004 message/cycle).**

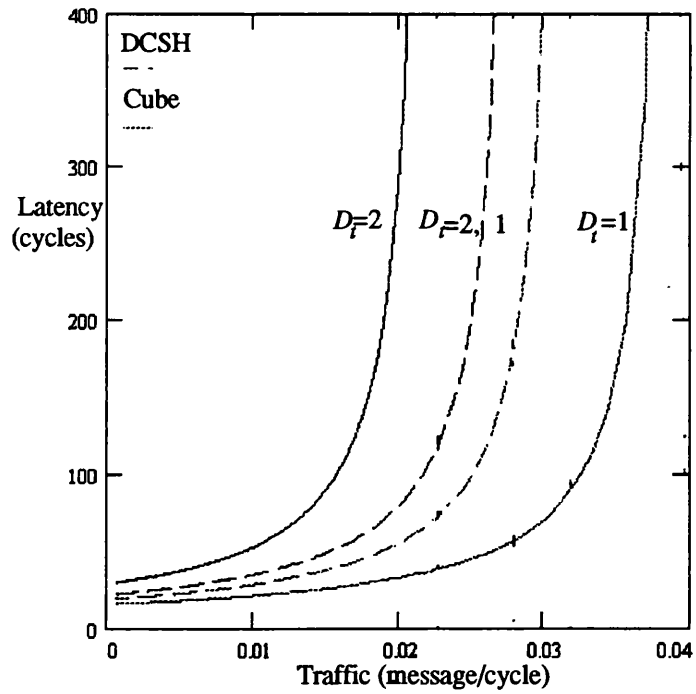



**Figure 6.22: Wormhole routing performance in the cube & 2-D DCSH for a fixed message length ($M$=16 phits).**

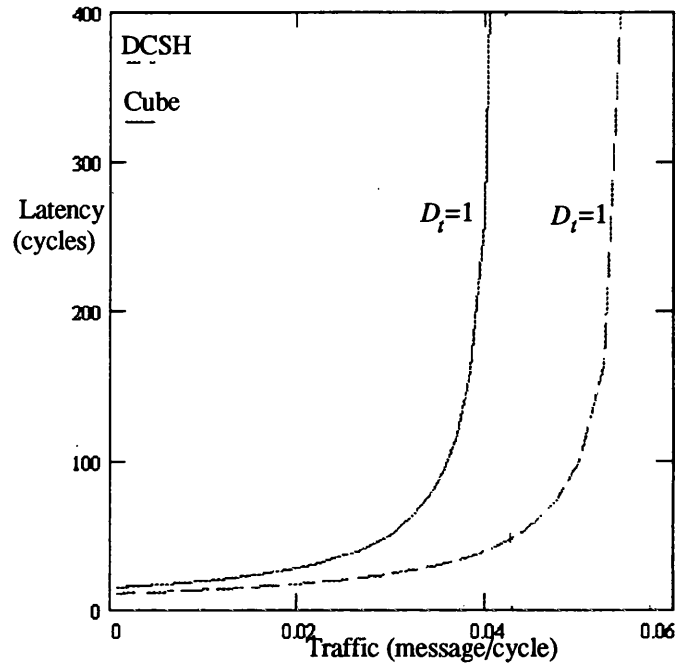



**Figure 6.23: Wormhole routing performance in the cube & 2-D DCSH for a fixed message length ($M$=8 phits).**

## 6.4 Conclusions

Low-dimensional DCSHs are more general structures than their MIN counterparts, as they are suitable for applications both with and without locality of communication. Under uniform traffic, low-dimensional DCSHs have comparable performance to that of their MIN counterparts of equal cost when implemented using either VLSI or multiple-chip technology. With communication locality, however, they provide better performance even for applications that contain only a small degree of locality.

Although low-dimensional DCSHs can have wider channels and therefore have a lower latency than their higher-dimensional counterparts by exploiting the layered implementation scheme, this chapter has also evaluated the performance merits of low-dimensional DCSHs when more conventional implementations are used. To this end, the 2-dimensional DCSH has been compared to the binary $n$-cube, widely used in first-generation multicomputers.

When implemented in VLSI, the binary $n$-cube has better performance than the 2-D DCSH at long message lengths with virtual cut-through. However, the cube must use the multiple-accepting model to take advantage of its rich interconnection structure at high

traffic. The binary $n$-cube is already an expensive topology to implement, and using this switching element model will make it even less appealing on cost grounds. At short message lengths the 2-D DCSH has a lower latency because the cube cannot use its large number of channels efficiently. With wormhole routing the 2-D DCSH has better performance than the binary $n$-cube because a message in the former goes through far less switching elements.

When implemented in multiple-chip technology, the binary $n$-cube outperforms the 2-D DCSH at long messages with virtual cut-through because of its wider channels. The advantage of the cube diminishes at short message lengths. The cube has better performance at long messages with wormhole routing, but for medium and short messages the 2-D DCSH is again superior. This reveals that the 2-D DCSH is more suitable for fine-grain computation and can take better advantage of high wiring densities and pin-out counts, which both favour short message aspect ratios.

# 7 Conclusions and future directions

Much recent research has been aimed at the development of multicomputer interconnection schemes which have desirable topological features, can easily exploit the available technology to overcome channel bandwidth constraints, and can readily employ effective switching methods and routing algorithms. This thesis has introduced a new class of regular multi-dimensional hypergraph networks, *distributed crossbar switch hypermeshes* (*DCSH*s), and has shown that low-dimensional structures in this class have considerable potential. The arguments for the use of the low-dimensional DCSH architecture can be summarised briefly.

- Multicomputers based on graph topologies such as meshes and tori are limited by simple interconnect patterns, and can ultimately only offer effective support to parallel applications that rely on strong communication locality. Hypermeshes, including DCSHs, provide more powerful and general structures, offer greater flexibility in the support of applications with non-local communication patterns, and permit effective emulation of other common topologies. Thus the DCSH supports applications that map well onto the mesh, torus, binary-tree and binary $n$-cube. It supports applications that map well onto multistage interconnection networks, such as the Omega and inverse Omega, and also applications which require broadcast operations.

- The provision of adequate channel bandwidth is the major obstacle in implementing most existing graph and hypergraph networks. In traditional network implementations, a node includes a processing element and switching element. However, this node-based partitioning of a topology is not necessarily the optimal way to use high wiring densities and pin-out counts. Low-dimensional DCSHs can take advantage of topological properties to separate switching from processing functions, increasing channel width (which might otherwise be expected to limit such a richly connected structure) and yet retaining a regular and physically compact form.

- The new layered implementation permits the DCSH to use moderately larger cluster sizes compared to previous schemes. For instance, cluster sizes of 32 and 64 nodes are entirely feasible. However, the crossbar switch hypermesh, which is the commonest and cheapest implementation, can use cluster sizes of no more than 16 nodes due to the bandwidth limitation of the commercially available crossbar switches.

- It has been shown that in addition to the two forms of the bandwidth constraint widely discussed in the literature, viz. wiring density and pin-out limits, a third, the relay bandwidth limit, is more relevant to layered implementations. Other graph and hypergraph networks are restricted in their ability to exploit structural layering due to the difficulty in providing the necessary relay bandwidth inside the switching elements; the low-dimensional DCSH escapes this difficulty due to the multiplexing-down function inherent in its switching elements.

- Sparsely-connected networks, such as the mesh and torus, cannot benefit from the multiple-accepting node model since the performance bottleneck resides in the network channel rather than at the processing element-network interface. In many richly-connected networks, such as the binary $n$-cube and generalised hypercube, if the multiple-accepting model is not used, the advantage of the large number of channels in the topology is lost, since the performance bottleneck is at the processing element-network interface. Yet the need for a multiple-accepting model makes these networks less appealing as it increases their implementation cost. In neither case does the network offer a connection structure that strikes a balance between the traffic capacity that the network can handle and that offered by the processing elements. Low-dimensional DCSHs can use the single-accepting model without any performance loss because it has the same latency as the multiple-accepting model.

This reduces implementation cost while still permitting the network to handle traffic capacity which is exactly matched to the capacity of the processing elements.

- When traffic load is approximately uniform, some networks (mesh, torus, binary $n$-cube) use restricted routing which provides better performance than random routing. However, in addition to being less fault-tolerant, restricted routing does not perform well in the presence of hot-spots. In low-dimensional DCSHs, however, random routing provides equivalent performance to restricted routing under uniform traffic load and so there is no disadvantage in employing it. This makes these networks less vulnerable to hot-spots and node failures.

- Many interconnection networks must use fast switching elements in order to reduce the impact of decision time on message latency, especially when wormhole routing is employed. The networks that suffer most are ones such as tori and meshes, which have large diameter even for relatively small network sizes. The decision time also causes performance degradation for the binary $n$-cube as the system size increases. Low-dimensional DCSHs can afford to use slower, and hence cheaper, switching elements with higher decision time without any deleterious effect on performance.

- Adaptive routing has the advantage of reducing the effects of congestion. However, a study by Chien *et al* has shown that these routing strategies tend to increase decision time due to the more complex circuitry of the switching elements [66]. For this reason it has been avoided in practical multicomputers where low decision time is critical. As discussed above, low-dimensional DCSHs are not subject to such considerations and could use adaptive routing effectively.

To further support the above considerations, the DCSH has been compared to other hypermeshes, such as spanning-bus hypercubes, crossbar switch hypermeshes, and generalised hypercubes. They have also been compared to the mesh and multi-stage interconnection networks, which have been widely used in message-passing multicomputers and shared-memory multiprocessors respectively. Finally, the relative merits of lower-dimensional DCSHs against their higher-dimensional counterparts (binary $n$-cubes) have been assessed. In all cases, realistic assumptions have been made, including the use of pipeline-bit transmission and non-zero decision times.

Under reasonable assumptions, the new proposed low-dimensional DCSHs generally have superior performance characteristics. With virtual cut-through, this is true under all constraints when messages are short. With wormhole routing, it is true under all constraints at all message lengths. The studies reveal that decision time is a very important factor when wormhole routing is used, as it can considerably alter the outcome.

The findings also illustrate two important characteristics of the DCSH architecture. Firstly, the DCSH is more efficient in taking advantage of high wiring densities or pin-out counts, when available. Secondly, the DCSH is especially well suited to fine-grain computation, which will become increasingly important as appropriate parallel programming techniques are developed.

For the foreseeable future, networks will continue to be laid out over multiple chips. Further, wormhole routing will continue to be a popular switching method since it reduces storage requirements considerably while coping with the moderate traffic generated by current coarse/medium-grain programming techniques. Virtual cut-through provides the best performance, at higher implementation cost. The DCSH provides superior performance when implemented in multiple-chip technology with wormhole routing.

There are number of issues and open problems that require further investigation. These are summarised below.

- The multiple-accepting model has clear performance benefits over the single accepting model for high-dimensional DCSHs, but is more costly to implement. The analyses presented above have assumed that the number of channels leading to the local processing element in the multiple-accepting model was equal to the network dimensionality. An important open problem would be to investigate the optimal number of these local channels. If a smaller number of local channels can achieve near-optimal performance, the implementation cost of the switching element can be considerably reduced. This issue is equally important for graph as well as other hypergraph networks.

- Several parallel applications are inherently suited to the shared-memory programming model. Recently, "shared memory" multicomputers (typically based on meshes and

binary $n$-cubes) have been proposed, as these can scale to larger sizes than those based on buses and multistage interconnection networks [4,46, 47, 135]. It seems likely that with the ability to handle large numbers of small messages, the DCSH would support this programming model particularly effectively, but research is required to confirm this and perform a comparative study with meshes and hypercubes. The analyses presented throughout this thesis have been confined to the network level, in that details relevant to issues, such as the nature of the programming model in use, have not been included. A more detailed analysis would include these features.

- Several comparative analyses of networks have used wiring density and pin-out count to quantify implementation cost in VLSI and multiple-chip technology. However, these do not take into account the cost of the switching elements which may be complex and therefore expensive to implement. In this thesis, a new concept, the relay bandwidth, has been introduced to reflect the complexity of the switching element. Proper quantifications for this are needed as well as a better understanding of its interaction with the other constraints, to yield more accurate and complete network performance studies.

- Hypermeshes are well-suited to applications that require broadcast operations. An open area of research is the development of efficient broadcast algorithms and mathematical models which quantify the suitability of these networks for such algorithms and compare their performance results against those of spanning-bus hypercubes, meshes, tori, and binary $n$-cubes.

- The impact of adaptive routing and load balancing on the performance of graph networks, including meshes, binary $n$-cubes, and generalised hypercubes, has been widely reported in the literature [73, 74, 146, 147, 148, 149, 150]. It would be interesting to evaluate these strategies for the DCSH.

- One recent commercial machine, the CM5 from the Thinking Machine Co. [151], is based on a hypertree, originally proposed by Leseirson [152]. This is a binary tree with channels that are wider as they are closer to the root. No work has yet been reported which compares hypermeshes to this network.

# References


1. C.L. Seitz, *Concurrent VLSI architectures*, IEEE Trans. Comp., Vol. C33, No 12, Dec. 84, pp. 1247-1265.

2. C.L. Seitz, *Concurrent architectures*, in VLSI and Parallel Computation, in R. Suaya and G. Birtwistle (Eds.), Morgan Kaufmann Publishers, Inc, 1990.

3. D.A. Reed *et al*, *Multicomputer networks: Message-based parallel processing*, MIT Press, 1987.

4. W.C Athas and C.L. Seitz, *Multicomputers: Message-passing concurrent computers*, *IEEE Computer*, Vol 21, No 8, Aug. 88, pp. 9-24.

5. W.C Athas, *Fine-grain concurrent computation*, TR:5242:87, Computer Science Department, CalTech, 87.

6. D. Agarwal and V. Janakira, *Evaluating the performance of multicomputer configurations*, IEE Computer, May 86, pp. 9-24.

7. W.J. Dally, *Fine-grain concurrent computers*, Proc. 3rd Symp. on Hypercube Concurrent Computers and Applications, ACM 1988.

8. W.J. Dally, *The architecture of a message-driven processor*, Proc. of the 14th ACM/IEEE Symp. Comp. Arch., June 87, pp. 189-196.

9. L.G. Valiant, *A bridging model for parallel computation*, Comm. ACM, Aug. 90.

10. L.G. Valiant, *A scheme for fast parallel computation*, SIAM J., Vol. 11, No. 2, May 82, pp 103-111.

11. S.B. Akers and B. Krishnamurthy, *A group-theoretic model for symmetric interconnection networks*, IEEE Trans. Comp., Vol C-38, No. 4, April 89.

12. C. Berge, *Graphs and hypergraphs*, North-Holland, 1973.

13. D.A. Reed, *Cost-performance bounds for multicomputers networks*, IEEE Trans. Comp., Vol C32, No 1, Jan. 834, pp. 83-95.

14. D.A. Reed and D.C. Grunwald, *The performance of multicomputer interconnection networks*, IEEE Computer, Vol 20, No. 6, June 87, pp. 63-73.

15. J. Ghosh and K. Hwang, *Hypernet: A multiprocessor interconnection topology*, IEEE Trans. Comp., Dec. 81, pp. 923-933.

16. S.P. Dandamudi, *On Hypercube-based hierarchical interconnection network design*, Journal of Parallel and Distributed Computing, Dec. 91, pp. 283-289.

17. D.A. Carlson, *Modified mesh connected parallel computer*, IEEE Trans. Comp., Oct. 88, pp. 1315-1321.

18. K. Efe, *A variation on the hypercube with lower diameter*, IEEE Trans. Comp., Vol C40, No. 11, Nov. 91, pp 1312-1316.

19. S. Abraham and K. Padmanabhan, *An analysis of the twisted cube topology*, 1989 Int. Conference on Parallel Processing, Aug. 89, pp. 116-120.

20. A. El-Amawy and S. Latifi, *Bridged hypercube networks*, Journal of Parallel and Distributed Computing, Oct. 90, pp 90-95.

21. C.L. Seitz et al, *Engineering limits on computer performance*, Physics Today, 84.

22. W.J. Dally, *A VLSI architecture for concurrent data structures*, Kluwer, 87.

23. S. Abraham and K. Padmanabhan, *Performance of multicomputer networks under pin-out constraints*, J. Par. & Dist. Systems, pp 237-248, Aug. 92.

24. M.A. Franklin, *Pin limitation and partitioning of VLSI interconnection networks*, IEEE Trans. Comp., Vol C36, No. 5, May 87, pp. 547-553.

25. G. Byrd and B. Delagi, *Consideration for multiprocessor topologies*, STAN-CS-87, 11484, Dept. Comp. Sci., Stanford University, Jan. 87.

26. P. Mazumder, *Evaluation of on-chip static interconnection networks*, IEEE Trans. Comp., Vol C36, No. 3, March 87, pp. 365-369.

27. E. Horowitz and A. Zorat, *The binary tree as an interconnection network: Applications to multiprocessor systems and VLSI*, IEEE Trans. Comp., Vol C-30,

No. 4, April 81, pp. 247-253.

28. S. Abraham and K. Padmanabhan, *Performance of the direct binary n-cube networks for multiprocessors*, IEEE Trans. Comp., Vol 37, No. 7, July 89.

29. T. Szymanski and C. Feng, *Design and analysis of buffered crossbars and Banyans with cut-through switching*, Int. Conf. Parallel Proc. 90, pp 264-273.

30. C. Wu and M. Lee, *Analysis of multistage interconnection network configurations and operations*, IEEE Trans. Comp., Vol. 41, No. 1, Jan. 92, pp 18-27.

31. H. Yoon and Y. Lee, *Performance analysis and comparison of packet switching interconnection networks*, Int. Conf. Parallel Processing, 86, pp 542-545.

32. S.P. Dandamudi and D. L. Eager, *Hierarchical interconnection networks for multicomputers systems*, IEEE Trans. Comp., Vol. 39, June 89, pp 653-660.

33. K. Padmanabhan, *Design and analysis of even-sized binary shuffle-exchange networks for multiprocessors*, IEEE Trans. Parallel & Distributed Systems, Vol. 2, No. 4, Oct. 91.

34. S. Abraham, *Issues in the architecture of direct interconnection schemes for multiprocessors*, Ph.D. thesis, Univ. of Illinois at Urbana-Champaign, 90.

35. R.W. Hockney and C.R. Jesshope, *Parallel computers 2*, Adam Hilger 88.

36. C.D. Thomson, *A complexity theory of VLSI*, Ph.D. thesis, Technical Report CMU-CS-80-140, Dept. of Computer Science, Carnegie-Mellon Univ., Aug. 80.

37. W.J. Dally, *Performance analysis of k-ary n-cubes interconnection networks*, IEEE Trans. Comp., Vol. 39, No. 6, June 90.

38. W.J. Dally, *Network and processor architecture for message-driven computers*, in R. Suaya and G. Birtwistle (Eds.), Morgan Kaufmann Publishers, Inc, 1990.

39. Intel Scientific Systems, *Intel iPSC users guide, Intel Corporation*, Beaverton, Oregon, Order Number: 175455-003.

40. R. Arlanskas, *iPSC/2 system: A second generation hypercube*, Proc. 3rd ACM

Conf. Hypercube Concurrent Computers and Applications, Vol 1, Jan. 88.

41. N-Cube Systems, *N-cube Handbook*, N-Cube, 1986.

42. C.L. Seitz, *The Cosmic Cube*, Comm. ACM, Vol 28, Jan. 85.

43. W.J. Dally et al, *The J-machine: A fine-grain concurrent computer*, Information Processing 89, Elsevier, 89, pp. 1147-1153.

44. C.L. Seitz *et al*, *The architecture and programming of the Ametek Series 2010 multiprocessor*, Proc. 3rd Conf. Hypercube Concurrent Comp. and Appl., Jan. 88.

45. C. Peterson *et al*, *iWarp: a 100-MOPS LIW microprocessor for multicomputers*, IEEE Micro, Vol. 11, No. 13, June 91.

46. D. Lenoski, J. Laudon, K. Gharacholoo, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, *The Stanford DASH multiprocessor*, IEEE Computer, Mar. 92.

47. J.T. Kuehn and B.J. Smith, *The HORIZON supercomputing system: Architecture and software*, Proc. Supercomputing 88, Nov. 88.

48. J. Rattner, *The new-age of supercomputing*, Lecture notes in Computer Science, Arndt Bode (ed.), Springer-Verlag, pp 1-6, April 91.

49. S. Abraham and K. Padmanabhan, *Constraint-based evaluation of multicomputer networks*, Int. Conf. Parallel Processing, 1990.

50. L.M. Mackenzie, M. Ould-Khaoua, R.J. Sutherland, and T. Kelly, *The COBRA project: alleviating bandwidth constraints in large multicomputer networks*, Proc. Supercomputing Symposium 92, Canada, pp 15-28, June 92.

51. S.L. Scott and J.R. Goodman, *Performance of pipelined-channel k-ary n-cube networks*, Preprint, Computer Science Department, University of Wisconsin, 1992.

52. A. Agarwal, *Limits on interconnection network performance*, IEEE Trans. Parallel and Dist. Syst., Vol. 2, No. 4, Oct. 91.

53. T. Kelly, L.M. Mackenzie and M. Ould-Khaoua, *Effect of message length, decision time and wiring density on performance of k-ary n-cube latency*, Permian

Basin Supercomputing Conference, Texas, March 92.

54. W.T. Hsu and P. Yew, *The impact of wiring constraints on hierarchical network performance*, IEEE Symposium on Parallel and distributed processing, 1992.

55. W.J. Dally, *Express Cubes: Improving the performance of k-ary n-cube interconnection networks*, IEEE Trans. Comp., Vol. 40, No. 9, Set. 91.

56. C.W. Athas, *Physically compact, high-performance multicomputers*, Advanced Research in VLSI, MIT Press, 1990, pp. 179-192.

57. M. Ould-Khaoua, L.M. Mackenzie, R.J. Sutherland, & T. Kelly, *Hypergraph-based interconnection network for large multicomputers*, CONPAR-VIPP 92 Proceedings, Lyon, France, appeared in Lecture Notes in Computer Science, Springer-Verlag Eds., Sept. 92.

58. A.S. Tanenbaum, *Computer networks*, Prentice-Hall Int., Inc. (2nd Ed.), 1989.

59. P. Kermani and L. Kleinrock, *Virtual cut-through: A new computer communication switching technique*, Comp. Networks, Vol. 3, pp 267-286, 79.

60. C.L. Seitz, *The hypercube communication chip*, Dep. Comp. Sci., CalTech, Display File 5182:DF:85, March 85.

61. E. Chow *et al*, *Hyperswitch network for the hypercube computer*, Proc. ACM 15th Int. Symp. on Comp. Arch., pp 90-99, 91

62. S.F. Neugent, *The iPSC/2 direct-connect communication technology*, Proc. Hypercube Concurrent Computers and Applications, 1988, pp 51-60.

63. P. Kermani and Kleinrock, *A Trade-off study of switching systems in computer communication networks*, IEEE Trans. Comp., Vol. 29, No. 12, 1980.

64. W.J. Dally and P. Song, *Design of a self-timed VLSI multicomputer communication controller*, Proc. Int. Conf. Computer Design, IEEE CS Press, Los Alamitos, Calif., Order No. 2473, 1987, pp 230-234

65. W.J. Dally and C.L. Seitz, *The torus routing chip*, J. Distributed Computing, Vol.1, No. 3, pp 187-196, 86.

66. A.A. Chien, *A cost model for k-ary n-cubes wormhole routers*, Preprint, Department of Computer Science, University of Illinois at U-C, 1993.

67. W.J. Dally *et al*, *The message-driven processor: A multicomputer processing node with efficient mechanisms*, IEEE Micro 92.

68. M. Noakes and W.J. Dally, *System design of the J-machine*, Advanced Research in VLSI, MIT Press, pp 179-192, 90.

69. D. May, *The next generation transputer and beyond*, Lecture notes in Computer Science, Arndt Bode (ed.), Springer-Verlag, pp 7-12, April 91.

70. L.M. Ni and P.K. McKinley, *A survey of wormhole routing techniques in direct networks*, IEEE Computer, pp 62-76, Feb. 93.

71. P.T. Gaughan *et al*, *Adaptive routing protocols for hypercube interconnection networks*, IEEE Computer, May 93.

72. G.J. Glass and L.M. Ni, *The turn model for adaptive routing*, Proc. 19th Int. Symp. on Computer Architecture, pp 278-287, 92.

73. S. Kanstantinidou, *Adaptive, minimal routing in hypercubes*, Proc. 6th M.I.T. Conf. Advanced Research in VLSI, MIT Press, pp 139-153, 90.

74. J.Y Ngai *et al*, *A Framework for adaptive routing in multicomputer networks*, Ph.D. thesis, Computer Science Dept., CalTech, 89.

75. J.P. Hayes *et al*, *A Microprocessor-based hypercube supercomputer*, IEEE Computer, pp 7-17, Oct. 86.

76. K. Aoyama, *Design issues in implementing an adaptive router*, M.S. Thesis, Comp. Sci. Dept., Texas Univ., Jan. 93.

77. D. Gelernter, *A DAG-based algorithm for prevention of store-and-forward deadlock in packet networks*, IEEE Trans. Comp., Vol. 30, No. 10, Oct. 81

78. P.L. Merlin and P.J. Schweitzer, *Deadlock avoidance-store-and-forward deadlock*, IEEE Trans. Comm., Vol. 28, pp 345-355, March 80.

79. P.M. Merlin and P.J. Schwietzer, *Deadlock avoidance in store-and-forward networks II - other deadlock types*, IEEE Trans. Comm., Vol 28, Mar. 80.

80. K.D. Gunther, *Prevention of deadlock in packet-switched data transport systems*, IEEE Trans. Comm., Vol. 29, No. 4, pp 512-524, April 81.

81. W.J. Dally, *Deadlock-free message routing in multiprocessor interconnection networks*, IEEE Trans. Comp., Vol.C-36, No. 5, pp 547-553, May 87.

82. D.H. Linder and J.C. Harden, *An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes*, IEEE Trans. Comp., Vol. 40, No. 1, pp 2-12 Jan. 91.

83 H. Sullivian and T.R. Barkshow, *A large scale homogeneous machine*, Proc. ACM 4th Annu. Symp. on Comp. Arch., pp 105-124, 79.

84. K. Hwang and F. Briggs, *Computer architecture and parallel processing*, McGraw-Hill International Eds., 1984.

85. G.S. Almasi and A. Gottlieb, *Highly parallel computing*, The Benjamin/Cummings Publisher Company, Inc., 1989.

86. H.J. Seigel, *Interconnection networks for SIMD machines*, IEEE Comp. Mag., Vol. 12, No. 6, pp. 57-65, June 79.

87. T. Szymanski, *A fiber optic hypermesh for SIMD/MIMD machines*, 1990 int. Conf. on Parallel Processing, pp. 710-718.

88. A.B. Ruighaver, *A decoupled multicomputer with full interconnection*, CONPAR-VIPP 92 Proceedings, Lyon, France, appeared in Lecture Notes in Computer Science, Springer-Verlag Eds., Sept. 92.

89. Z. Guo *et al*, *Pipelined Communication in optically interconnected arrays*, Journal of Parallel and Distributed Computing, Aug. 91, pp. 269-282.

90. W.K. Giloi *et al*, *Choosing the interconnect of distributed memory systems by cost and blocking behaviour*, IEEE Parallel Processing Symp., 1991, pp. 438-444.

91. P.W. Dowd and K. Jabbour, *Spanning multi-access channel hypercube computer interconnection*, IEEE Trans. Comp., Vol. C-37, No. 9, Sept. 88, pp. 283-290.

92. N. Tanabe *et al*, *Base-m n-cube: High performance interconnection networks for highly parallel computer PRODIGY*, 1991 Int. Conf. on Parallel Processing, Aug. 91, pp. 509-516.

93. H. Cha *et al*, *The hyperdynamic architecture (massively parallel message-passing machine) -architecture and performance*, Int. Conf. Supercomputing, June 91.

94. T. Feng, *A survey of interconnection networks*, IEEE Computer, Dec.81.

95. S.H. Bokhari, *Finding the maximum on an array processor with a global bus*, IEEE Trans. Comp., Vol C32, No 9, 9180, pp. 836-840.

96. K.V. Prasanna *et al*, *Array processor with multiple broadcasting*, J. Parallel & Distributed Computing, March 87, pp. 173-190.

97. K.V. Prasanna *et al*, *Image computations on meshes with multiple broadcast*, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-11, No. 11, 1989, pp. 1194-1202.

98. S. Chittor *et al*, *Performance degradation in large wormhole routed inter processor communication networks*, Int. Conf. on Parallel Processing, 90.

99. S. Chittor *et al*, *Performance evaluation of mesh-connected wormhole routed networks for interposes communication in multicomputers*, Int. Conf. on Parallel Processing, 90.

100. D.M.N. Prior *et al*, *What Price regularity?*, Concurrency: Practice and Experience, Vol 2, No. 1, March 90, pp 55-57.

101. Y. Ofek, *The topology, algorithms and analysis of a fiber optic hypergraph network*, Ph.D. thesis, Elec. Eng. Dept., Univ. Illinois-Urbana, 87.

102. T. Szymanski, *O(LogN/LogLogN), randomised routing in hypermeshes*, 1992 . Int. Conf. Parallel Processing, pp 710-718.

103. L.M. Mackenzie *et al*, *COBRA: A high-performance interconnection network for large multicomputers*, Technical-Report 119/R19, Comp. Sci. Dept., Univ. of Glasgow, Oct. 91.

104. Eva Ma and Lixin Tao, *Embedding among meshes and tori*, Tec-Rep. MS-CIS-88-63, Univ. of Pennsylvania, Aug. 88.

105. G.C. Fox *et al*, *Solving problems on concurrent processors*, Prentice Hall, 1988.

106. G.A. Giest *et al*, *Parallel Algorithms for matrix computation*, The Characteristics Of Parallel Algorithms, Eds. L.H. Jameison *et al*, MIT Press, 1987.

107. L.D. Wittie, *Communication structures for large networks of microcomputers*, IEEE Trans. Comp., Vol. C-30, No. 4, 1981, pp 264-273.

108. D.B. Gustavson, *Wire-Or logic on transmission lines*, IEEE Micro, June 83.

109. L.N. Bhuyan *et al*, *A general class of processor interconnection strategies*, Proc. 9th Int. Symp. Comp. Arch., April 82.

110. L.N. Bhuyan *et al*, *Generalised hypercube and hyperbus structures for a computer network*, IEEE Trans. Comp., Vol C-33, No. 4, April 84

111. P.D. Dowd, *High performance interprocess communication through optical division wavelength division multiple-access channels*, 18th Int. Symp. Comp. Arch., May 91.

112. J.E. Midwinter and M.G. Taylor, *The reality of digital optical computing*, IEEE LCS Magazine, May 90, pp. 179-192.

113. Y. Saad and M. Schultz, *Topological properties of hypercubes*, IEEE Trans. Comp., Vol C-37, No. 7, July 88.

114. Y. Saad *et al*, *Data communication in parallel architectures*, Parallel Computing, Vol 11, 89, pp. 131-150.

115. Bakoglu, *Circuits, interconnections and packaging for VLSI*, Addison-Wesley, 90.

116. O. Kwon and R.F.W. Pease, *Closely packed micro-strip lines as very-high speed chip-to-chip interconnects*, IEEE Trans. Component, Hybrids and Manufacturing Technology, Vol 10, No. 3, Sept. 87.

117. C.L. Seitz, *System timings in introduction to VLSI systems*, C. Mead and L.

Conway, Addison-Wesley, 1980.

118. S. Harris and S. Radley, *The design and manufacture of HDMI*, Nepcon Europe, Mar. 91.

119. K. Johnson, *The impact of communication locality on large-scale multiprocessor performance*, Proc. of 19th Annual International Symp. on Comp. Arch., May 92.

120. S.V. Dandamudi, *Hierarchical interconnection networks for multicomputer systems*, Ph.D. thesis, Department of Computational Science, University of Saskatchewan, Saskatoon, Canada, Nov. 88.

121. L.M. Mackenzie *et al*, *COBRA: A high-performance interconnection network for multicomputers*, Proceedings of the IEE Colloquium on Medium-Grain Distributed Computing, London, April 92.

122. C.P. Kruskal and M. Snir, *The performance of multistage interconnection networks for multiprocessors*, IEEE Trans. Comp., Vol. 32, pp. 1091-1098, Dec.83.

123. L. Kleinrock, *Queueing systems*, Vol. 1, John Wiley and Sons, New York, 75.

124. K. Padmanabhan, *On the trade-off between node degree and communication channel width in shuffle-exchange networks*, 3rd IEEE Symp. on Parallel and Distributed Processing, Dec. 91.

125. H.M. Dougal, *Simulating computer performance: Techniques & tools*, MIT Press, 87.

126. CACI Co., *Introduction to simulation using SIMSCRIPT*, 84.

127. G.J. Pfister and V.A. Norton, *Hot-spot contention and combining in multistage interconnection networks*, IEEE Trans. Comp., Vol C-34, Oct. 85.

128. V.S. Adve and M.K. Vernon, *Performance analysis of multiprocessor mesh interconnection networks with wormhole routing*, Preprint, Computer Science Dept., University of Wisconsin-Madison, 1992.

129. D.C. Winsor nad T.N. Mudge, A*nalysis of bus hierarchies for multiprocessors*, 15th Annual Int. Symp. Comp. Arch., May 88.

130. S. Chittor et al, *Hypercubes vs. meshes*, Tech-Rept. CPS-89-17, Nov. 89.

131. J.Y. Hui and E. Arthurs, *A Broadband packet switch for integrated transport*, IEEE on Selected Areas in Communications, VOL. SAC-5, No. 8, Oct. 87.

132. W.B. Wike *et al*, *Alpha structure based B-HIVE multicomputer*, 3rd Conf. Hypercube Concurrent Computer and Applications, Jan. 88.

133. D.P. Agarwal *et al*, *B-HIVE: A heterogeneous, interconnected, versatile and expandable multicomputer system*, Computer Architecture News, Jan. 86.

134. R.E. Kessler, *CRAY T3D: A new dimension for Cray research*, Proc. IEEE Int. Conf. Supercomputing, 1993.

135. A. Agarwal et al, *The MIT Alewife machine: A large-scale distributed-memory multiprocessor*, LCS-Tech Memo. 454, MIR, Laboratory for Computer Science, 91.

136. D. May, *The next generation of transputers and beyond*, Lecture Notes in Computer, Arntd Bode (ed.), Springer-Verlag, April 91.

137. L.V. Kale, *Optimal communication neighbourhoods*, Proc. Int. Conf. Parallel Processing, 86, pp 823-826.

138. T. Szymanski, *The complexity of FFT and related butterfly algorithms on meshes and hypermeshes*, 1992 Int. Conf. Parallel Processing.

139. D. Gajski, D. Kuck, D. Lawrie, and A. Saleh, *Cedar-A large scale multiprocessor*, Proc. of Int. Conf. on Parallel Processing, pp.. 524-52, Aug. 83.

140. G.F. Pfister *et al*, *The IBM Research Parallel Processing Prototype (RP3): introduction and architecture*, Proc. of Int. Conf. on Parallel Processing, pp. 764-771, Aug. 85.

141. T.F. Knight Jr., *Technologies for low latency interconnection switches*, Proc. of ACM Symp. on Parallel Algorithms and Architectures, June 89.

142. A. Gottlieb *et al*, *The NYU Ultracomputer-Designing a MIMD shared-memory parallel machine*, IEEE Trans. on Comp., Vol. C-32, Feb. 83.

143. J. Patterson *et al*, *The Mark III hypercube ensemble concurrent computer*, Proc. Int. Conf. Parallel Processing, 85.

144. Wu and Feng, *On a class of multistage interconnection networks*, IEEE Trans. Comp., Vol. 28, No. 8, pp 694-702, 80.

145. P. Wiley, *A parallel architecture comes of age at last,* IEEE Spectrum, June 87.

146. P.T. Gaughan and S. Yalamanchili, *Adaptive routing protocols for hypercube interconnection networks*, IEEE Computer, May 93.

147. S. Young and T. Yalamanchili, *A taxonomy and adaptive routing protocols for generalised hypercube architectures*, TR-GIT/CSRL-91/03, Comp. Sci. Dept., Georgia Institute of Technology, 91.

148. R.V. Hanxleden, *Load balancing on message passing architectures*, J. Parallel & Distributed Computing, Vol 13, 1991.

149. F.C. Lin *et al*, *The gradient model load balancing method*, IEEE Trans. Software Engineering, Vol. SE-13, No. 1, Jan. 87.

150 D.C. Marinescu *et al*, *The effects of Communication latency upon synchronisation and dynamic load balance on a hypercube*, Int. Conf. Parallel Processing, 1991.

151 Thinking Machines Corporation, Cambridge, MA. The Connection Machine CM-5 Technical Summary, January 92.

152 L.V. Leiserson, *Fat Trees: Universal networks for hardware-efficient supercomputing*, IEEE Trans. Comp., Vol. C-34, No. 10, Oct. 85.

## Publications during this work

1. *Hypergraph-based interconnection network for large multicomputers*, CONPAR-VIPP 92 Proceedings, Lyon, France, appeared in Lecture Notes in Computer Science, Springer-Verlag Eds., Sept. 92.

2. *The COBRA Project: alleviating the bandwidth constraints in large multicomputer networks*, Supercomputing Symposium 92 Proceedings, Montreal, Canada, June 92.

3. *COBRA: A high-performance interconnection network for multicomputers*, Proceedings of the IEE Colloquium on Medium-Grain Distributed Computing, London, April 92.

4. *The effect of message length, decision time and wiring density on k-ary n-cubes performance*, Permian-Basin Conference, Texas, March 92.

5. *COBRA: A high-performance interconnection network for large multicomputer*, Tech-Rep. R119/91, Comp. Sci. Dept., Glasgow Univ., Oct. 91.

6. *The performance of the switching methods in COBRA networks*, Tech-Rep. AH-93-01, Comp. Sci. Dept., Glasgow Univ., Nov. 93.

7. *Performance comparison of hypergraph and graph networks*, submitted at the International Conference on Massively Parallel Processing, Netherlands, June 94.