

**Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596**

COMMUNICATIONS ON VLSI

Frank Andre Schaffa

**July 1990
CSD-900019**

UNIVERSITY OF CALIFORNIA

Los Angeles

Communications on VLSI

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Frank Andre Schaffa

1989

© Copyright by

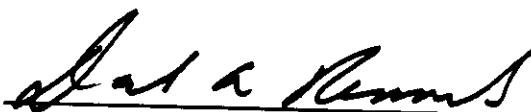
Frank Andre Schaffa

1989

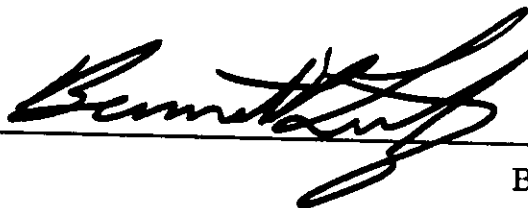
The dissertation of Frank Andre Schaffa is approved.



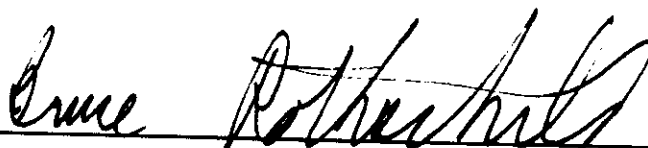
Gerald Estrin



Dave A. Rennels



Bennet Lientz



Bruce Rothschild



Mario Gerla, Committee Chair

University of California, Los Angeles

1989

In memory of my grandmother Klara

To my wife Vera

To my parents Alfred and Ellen

To my brothers Ralph, Thomas, and Ronald

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Related Works	2
1.3	Contributions and Organization of this Dissertation	3
2	Interconnection and Delay	5
2.1	Introduction	5
2.2	Components	6
2.2.1	Scaling	8
2.3	Area and Interconnection	10
2.4	Delay	11
2.4.1	Driving Schemes	11
2.4.2	Results	13
2.4.3	Delay and Throughput	15
2.5	Conclusions	18
3	Network Choices for VLSI	20
3.1	Introduction	20
3.2	Taxonomy	20

3.3	Performance Criteria	22
3.4	Design Criteria	23
3.5	Critique of some Topologies for VLSI	24
3.5.1	Discussion of Solutions	25
3.6	“Appropriate” Networks for VLSI	27
4	VLSI Networking Approach	29
4.1	Introduction	29
4.2	Motivation	29
4.3	Networking	31
4.4	Differences from Conventional Environments	32
5	The Grid/RingNet Solution Approach	34
5.1	GridNet	34
5.1.1	Topology	34
5.1.2	Protocol	35
5.1.3	Results - GridNet	38
5.2	GridNet/p2 - A Different Protocol	42
5.2.1	Protocol	43
5.2.2	Results	45
5.3	RingNet	49
5.3.1	<i>2in - 2out</i> Topologies	49
5.3.2	Protocols for Slotted Rings	52
5.3.3	Protocols for RingNet	66
5.3.4	Results	70
5.4	Conclusions	72

6 Routing	75
6.1 Our Domain	75
6.2 Routing Mechanisms	76
6.2.1 Broadcasting	80
6.3 The Routing Problem	82
6.4 Routing Algorithm	83
6.5 Applying the Results	85
6.6 Routing Assignment by Flow Deviation	87
6.7 Comparing the Algorithms	88
6.8 Conclusions	91
7 Conclusions	92
7.1 Summary of Results	92
7.2 Extensions of this Work	93
Bibliography	96

List of Figures

2.1	Geometry of Interconnection	6
2.2	Geometry of a MOS Transistor	7
2.3	Interconnection Scaling	9
2.4	RC Model	11
2.5	Driving Schemes	14
2.6	Delay versus Distance	15
2.7	Zoning of a Interconnection Wire	16
2.8	Throughput and Delay versus Number of Segments	17
2.9	G versus Number of Segments	18
3.1	Topologies Classification	22
3.2	Part of the Network Mesh and Timing	27
4.1	The Conceptual Hierarchy of VLSI Chip	30
4.2	The Scaling of the Hierarchy of a VLSI Chip	31
5.1	GridNet	35
5.2	Timing of Frames	36
5.3	Frame Generator	36
5.4	The Switch	37
5.5	The Line Handler	37

5.6	Delay versus Arrival Time for GridNet	39
5.7	Throughput versus Interarrival Time for GridNet	39
5.8	Throughput per Module versus Interarrival Time for GridNet	40
5.9	Throughput per Link versus Interarrival Time for GridNet	40
5.10	Delay versus Interarrival Time	45
5.11	Throughput versus Interarrival Time	46
5.12	Delay versus Throughput	47
5.13	Total Throughput versus Packet Interarrival Time	47
5.14	Power versus Packet Interarrival Time	48
5.15	Coefficient of Variation versus Packet Interarrival Time	48
5.16	MRing Topology	50
5.17	DRing Topology	51
5.18	Cubic Topology	52
5.19	Cyclic Topology and Flow	54
5.20	Timing and Flow of Data and Control	54
5.21	Flow of Free Frame Requests	56
5.22	One Buffer Only	57
5.23	Progression of Packets in a Lock Step Case	60
5.24	<i>RFW</i> and <i>DW</i> Windows	62
5.25	Two Buffers Only	63
5.26	Delay versus Packet Interarrival Time	65
5.27	Throughput versus Packet Interarrival Time	66
5.28	Flow of Data in a Switch	67
5.29	Node Traffic Flow	69
5.30	Delay versus Packet Interarrival Time	71

5.31	Throughput versus Packet Interarrival Time	71
5.32	Delay versus Packet Interarrival Time	72
5.33	Throughput versus Packet Interarrival Time	73
5.34	Delay versus Packet Interarrival Time	73
5.35	Throughput versus Packet Interarrival Time	74
6.1	Network Topology	76
6.2	Deflection Routing	77
6.3	Routing with Table-Lookup (RT)	79
6.4	Routing with Shift and Decide (SD)	80
6.5	Comparison Between Fixed and Deflection Routing	81
6.6	Broadcast Originating from <i>switch</i> ₁₀	82
6.7	Total Delay versus Interarrival Time for Initial and Optimal RT	86
6.8	Total Throughput versus Interarrival Time for Initial and Optimal RT	86
6.9	Power versus Interarrival Time	87
6.10	Total Time versus Total Throughput for Uniform Load	89
6.11	Total Time versus Total Throughput for Non Uniform Load	90

List of Tables

2.1	MOS Device Scaling	8
2.2	Interconnection Scaling	9
2.3	State of the Art Technology Evolution	10
3.1	Time Comparison Between Unidirectional and Half-Duplex Links	27
5.1	Aggregated Link Utilization Efficiency	49
5.2	Decision Logic on $Plane_x$	67
6.1	Decision Logic for Output Links	78
6.2	Table Entries for Broadcast Originating from $switch_{10}$	81
6.3	Distribution of Alternative Minpaths Between Source-Destination Pairs	84

ACKNOWLEDGMENTS

The Ph.D. process is long and depends on the help and involvement of many people. I am very thankful to those who have helped me along the way.

First of all, I would like to thank my advisor and friend Mario Gerla for his support in those difficult moments and for always being there when I needed him. He was patient and forthcoming with directions and suggestions for improvements to my research. I am also thankful for the other committee members, Gerald Estrin, Dave Rennels, Bennet Lientz, and Bruce Rothschild.

During my stay at UCLA, I have made friends for life, thus fulfilling needs other than academic ones. I would like to thank my brazilian friends, Edmundo, Valmir, Rolim, Nagib, and Suruagy, other fellow students, Jaime, Miquel, Ravi, Dorab, Paul, Marc, Hector, and Leon. I thank Dick Muntz for many interesting and in depth discussions on many different subjects. Verra Morgan has been an incredible friend, that I so often depended on. I thank Doris Sublette for her helpful and positive attitude. Finally I thank the dissertation group for many lively discussions on this subject matter.

I am also grateful to the CAD-LAB folk that provided me with many hours of cpu time to run my RESQ simulations and experiments.

This work was partially supported by a fellowship from CAPES, from the Ministry of Education, Brazil. I am very thankful to them for making it possible for me to come to UCLA for the Ph.D. program.

Finally I would like to thank my parents, brothers, and especially my wife Vera for her encouragement, support, and for being so proud of my achievements.

VITA

- January 15, 1955 Born in Sao Paulo, Brazil
- 1977 *Engenheiro Eletricista*
Escola Politecnica da USP, Sao Paulo, Brazil
- 1979–1980 Teaching Assistant
Escola Politecnica da USP, Sao Paulo, Brazil
University of California, Los Angeles
- 1980 Masters of Science in Computer Science
Escola Politecnica da USP, Sao Paulo, Brazil
- 1981–1985 CAPES/Brazil Fellowship
- 1982–1989 Post Graduate Research Engineer
University of California, Los Angeles

PUBLICATIONS AND PRESENTATIONS

1. F. Schaffa and L. Moscato, "A Distributed Architecture Machine: Recovery through Reconfiguration," Fifteenth International Symposium on Mini and Microcomputers, April 1981.
2. F. Schaffa, "Reconfiguration on a Distributed Architecture Machine," EPUSP Master's Thesis, November 1980.

ABSTRACT OF THE DISSERTATION

Communications on VLSI

by

Frank Andre Schaffa

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1989

Professor Mario Gerla, Chair

As technology continues to evolve, we have reached a point where the internal interconnection in VLSI requires a more careful analysis due to the increasing delay of the interconnection in relation to gate delays. In the foreseeable future, the impact of cross chip communication will become increasingly more critical on the performance of integrated systems and the way they are designed.

In this work we propose an approach called RingNet, which is based on strategies that minimize the interconnection delay and at the same time increase the total throughput. RingNet is specifically tailored to the given design constraints, which are quite different from conventional network constraints. Various versions of the RingNet protocols are discussed in terms of different characteristics such as liveness, flow control, fairness, and buffer requirements. Results based on simulation are presented. These protocols have no central control, e.g., the access mechanism is distributed with inherent flow control.

An important issue in the performance of a network is how information is routed from source to destination. We develop a heuristic algorithm which achieves good results when compared with optimal mathematical programming solutions. We also developed approximate analytic solutions which we compare with simulation results.

The results obtained in this work can be extended to high speed packet switching networks, that integrate voice, video and data communication. The thrust is the use of streamlined protocols with low overhead to achieve low packets delay and overall high throughput.

Chapter 1

Introduction

1.1 Motivation

In the foreseeable future, because of the advances of technology and the continuous evolution of VLSI, we will be confronted by a new design paradigm as the performance bottleneck continues to shift from gate delay to interconnection delay. Therefore, we are reaching a point where the internal interconnection requires a more careful analysis and new directions for its design have to be sought.

The basic function of interconnection is to carry information between the processing elements of a system. Interconnection can be observed from the lowest level where it is just a piece of conducting material to a higher level where it encompasses a whole complex system. At the lowest level, wires connect gates, which are the lowest “processing” elements. The higher level, on chip, deals with the interconnection of modules and keeps evolving with the developing technology. The major impact of this new design paradigm will be reflected on the higher levels of the interconnection hierarchy.

Two major issues concerning interconnection in VLSI are 1) the cost in area [Sei84] and 2) the cost of propagation delay [YT84]. Interconnection is known to employ a great deal of area, which is a major resource. Interconnection competes with how much functionality can be integrated. The other issue, dealing with propagation delay of signals in the interconnection on chip is mainly caused by stray capacitance and resistance.

We will show that we can achieve some definite benefits in terms of interconnection sharing. Sharing can save area; however, this is only achieved by implementing some higher level of functionality, for instance, routing and destination checking. This is accomplished by additional logic that also takes area and power. There-

fore, for a given communication scheme (protocol, routing, node interconnection) area can be saved if compared with a point to point non-shared interconnection. As a result, this “saved area” can be used for increased integration of functions or for redundancy, which may increase reliability. Moreover, regarding reliability, the existence of a network makes it easier to functionally replace a non-working module.

1.2 Related Works

Another motivation for this work is the difficulty of implementing large clocked systems [FK85]. The reason is the inevitable problem of clock skews and delays, which can be especially acute in VLSI systems as they become larger and denser. The paper [FK85] shows that on two-dimension arrays, which is the case for VLSI, it is impossible to run a clock such that the maximum clock skew between two communicating modules will be bounded by a constant as the system grows, and therefore an asynchronous scheme is proposed. They also made some experimental trials from which they concluded that an equipotential clock distribution was many times slower than a pipelined one.

Anceau [Anc82] presents a synchronous solution based on a distributed system organization, where each module has its own fast clock (each module is within a small isochronic region). It communicates with the other modules also synchronously via a bus, however at a slow clock speed. Metastability problems are avoided by preserving the clocks in phase. The obvious drawback of this approach is the slow communication capability because of the speed of the bus and the contention for this single slow broadcast bus. This result reinforces our proposed system framework of independent modules that are interconnected to others via a network with the objective of optimizing throughput and minimizing delay. The network operates independently of the modules. This distinct functionality assists the hierarchical design approach, which is specially desirable in VLSI.

Mazumder in [Maz87] presents an interesting evaluation of three types of static interconnection for VLSI with the criteria of three orthogonal aspects: physical (chip area and dissipation), computational speed (message delay and message density), and cost (chip yield, operational reliability, and layout cost). The three topologies considered were: binary tree, cube connected cycles and two dimension meshes. The choice of these topologies was motivated by their wide interest in the literature, for their optimal VLSI layout in terms of the AT^2 metric [Tho80], and because they belong to different classes.

The results are based on $O(N)$, where N is the number of “processors”. The problem with this approach is that N for a VLSI chip is rather small, between 16

and 64 and therefore $O(N)$ results can be misleading, because constant and other factors are not taken into account. In reference to the computational aspects, the average message delay is based on the average message path length. The problem is that this is only true if there is no contention for the links, hence this is an optimistic lower bound for the delay. The average message density [Maz87], defined as $\frac{N \cdot D}{L}$ (where N is the number of processors, D is the average path length, and L is the total number links), can provide some guidance in terms of bottlenecks, however it is important to keep in mind that this is very dependent on the traffic pattern, which is not considered there.

Seitz [Sei84] discusses the topologies for concurrent VLSI architecture in very general terms based mostly on the algorithm communication requirements without going into the merits of VLSI constraints.

Tewksbury [TH88] describes a sparsely connected mesh network for a large number of cells (modules) ($N > 400$) for a massive parallel multicomputer. However, while this approach might be interesting for WSI and wafer to wafer systems, we believe it is not practical for VLSI due to the large amount of resources needed just for the interconnection.

1.3 Contributions and Organization of this Dissertation

In Chapter 2, we start by analyzing a simplified model which describes the behavior of signals transmitted on long distance wires. This model is based on results taken from the existing literature and provides us with a better understanding of the delay performance due to stray capacitances and resistances. It also gives us the opportunity to observe how future technology, and its scaling down of component sizes, will affect the interconnection. Based on this model, different driving schemes for long distance interconnection are evaluated. The scheme that provides the best results is based on providing enough current to charge/discharge the stray capacitance, and segmenting the wire with buffers, so that the wire resistance does not limit the charging/discharging current. An interesting property results from the segmentation, namely, the “pipelining” effect that occurs since buffers isolate one segment from the other. Hence, different information may reside in the line as it is being piped through, effectively increasing the throughput. These two factors, lower delay and the pipelining effect, motivate us to search for a networking solution for the long distance communication problem on chip.

Our objective in Chapter 3 is to place the VLSI interconnection network in the general framework of interconnection networks. We start by presenting a taxon-

omy, followed by performance criteria for VLSI. We also discuss other important design criteria related to the VLSI arena, for instance, layout restrictions and number of ports per node. Common topologies are analyzed and appropriate design decisions are discussed based on the taxonomy.

In Chapter 4, based on the list of candidates and the design criteria presented in the previous chapter, we look further into the specific requirements and characteristics of an on-chip network that is appropriate for the general VLSI design framework. We find that the most cost effective approach is a hierarchical design approach. The most significant advantage of this approach is the separation of the communication “details” from the modules, namely, computation, which may facilitate resource sharing, logical reconfiguration, reduce the pin count, and others. Considerations on the implementation lead us to assume a network characterized by a small number of switches, streamlined protocols, and small packet size (word wide, bit long). We also discuss the advantage of using unidirectional links.

Our major contribution is detailed in Chapter 5 where we present different topologies and protocols to implement the network on chip. We discuss mesh topologies (Gridnet) and ring topologies (RingNet). A variety of protocols suitable for these topologies are described and analyzed in terms of different characteristics such as liveness, flow control, fairness, and buffer needs. Results based on simulation are presented.

Another important issue which impacts network performance, is the routing of information from source to destination. In Chapter 6, we present different routing schemes: adaptive (based on deflection) and; static (source routing as well as destination routing). We also develop an algorithm based on heuristics to find static routing tables for a given traffic pattern. The algorithm achieved good results when compared with optimal solutions. The latter were obtained by routing assignment employing the flow deviation method. A side benefit of the optimal routing model is to provide throughput and delay estimates which can be compared with simulation results.

In chapter 7 we summarize our research accomplishments and point to different directions in which this work may be extended.

Chapter 2

Interconnection and Delay

2.1 Introduction

In this chapter, we will discuss one important problem that afflicts interconnections in VLSI design and is becoming more prevalent as technology advances: the time delay of signals transmitted over the interconnection. The “real estate” area required by the interconnection is also important and will be briefly discussed.

The technological advancements in integrated circuit design has scaled down the minimum feature size, thus increasing the number of devices that can be integrated in a given area. At the same time, it has scaled up the size of the chip that can be produced with reasonable yields. The main benefit is that more functionality can be integrated; however, this does not happen without some cost, as we shall see.

The ultimate number of devices that can be integrated in a chip is limited not only by feature size, but also by the area taken up by interconnection. Studies in the published literature [Sei84, MRRS84, Gam81, HMD78] show that as more functionality is integrated, the greater is the area devoted to the interconnection of functional modules.

In terms of time behavior, the scaling down of the feature size will shift the dominant contribution of delay from the device itself to the long distance interconnections (across the chip) [GMS87, BM85, SM82].

We first present the fundamental components of VLSI that are related to performance. Following, we discuss the scaling effects on components, on interconnection, and on the area for interconnection. Next, we study the time behavior of signals over metal wires and signal driving schemes for long wires. Finally, we discuss some tradeoffs between delay and throughput.

This will serve as the basis for our networking proposal. The study presented here will be the underlying work from which we will develop the framework that will motivate the integration of a network on chip.

2.2 Components

Integrated circuits (MOS technology) are created by the superposition of different layers of conducting and insulating materials on top of a substrate. The proper placement of these layers will create the transistors (the active elements) and their interconnection. Each layer has resistance and capacitance; inductance will be assumed negligible. The characteristics of these elements determine the performance of the system.

The interconnection is also built as a conducting layer separated by insulation. Hence, interconnection lines have resistance as well as capacitance. Above certain dimensions, the interconnection may have a great impact in performance, since the current driving capacity of the active devices is quite limited (because of size). The electric current determines how fast the total capacitance is charged and/or discharged for a change in the signal applied to the lines.

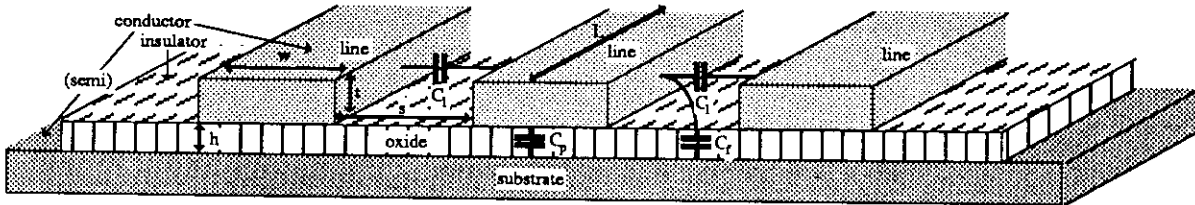


Figure 2.1: Geometry of Interconnection

We present now the electrical characteristics of the basic components:

- (a) *wires* - The geometry of an interconnection wire can be seen in Figure 2.1. The resistance of an interconnection (by unit length) can be expressed as:

$$R_I = \rho \frac{1}{wt} \quad (2.1)$$

where ρ is the resistivity of the conductor ($\rho = 2.7\mu\Omega \cdot cm$ Al), w t are respectively the width and the thickness of the interconnection.

The accurate evaluation of capacitance of the interconnection requires very time consuming calculations. Sakurai [Sak83] developed some simple formulas for wiring capacitance whose accuracy is within a few percent of more complex models. These results have been confirmed by [NDN87].

For our use, these approximations are quite acceptable. These results have also been used in other papers, for instance [BM84]. The capacitance per unit length includes the plate capacitance, the coupling capacitance between adjacent lines, and also the fringe capacitance. The expression for C_i per unit length is:

$$\frac{C_I}{E_{ox}} = 1.15\left(\frac{w}{h}\right) + 2.80\left(\frac{t}{h}\right)^{0.222} + 2\left[0.003\left(\frac{w}{h}\right) + 0.83\left(\frac{t}{h}\right) - 0.07\left(\frac{t}{h}\right)^{0.222}\right]\left(\frac{s}{h}\right)^{-1.34} \quad (2.2)$$

where E_{ox} is the dielectric constant for silicon oxide ($3.9 \cdot 8.85 \cdot 10^{-14} F/m$), h is the thickness of the oxide under the metal plate, s is the distance between adjacent metal lines, t and w are the thickness and the width respectively of the metal line. (Refer to Figure 2.1).

The first two terms in the expression are the dominant factors reflecting “area” and “fringe” capacitance. The other terms account for neighboring lines effect. Smith [SPP87] shows that the influence of added neighboring lines, for number of pairs ≥ 2 is rather small, and that the difference between one pair to two pairs is also quite small. This makes the 3 lines on ground plane model by Sakuray even more attractive.

- (b) *transistors* - Although transistors are non linear devices, we will apply a simple model that is used throughout the literature. [WE85, Muk85]

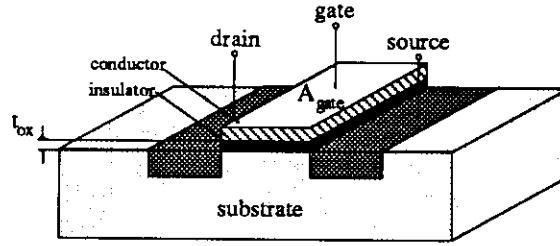


Figure 2.2: Geometry of a MOS Transistor

For capacitance,

$$C_{tr} = \frac{E_{ox}}{t_{ox}} \cdot A_{gate}$$

where E_{ox} is the permittivity of silicon oxide, t_{ox} is the the oxide thickness, and A_{gate} is the area of the gate.

For resistance,

$$R_{tr} = R_{\square} \cdot \frac{L}{W}, \quad R_{\square} = \left[\mu \frac{E_{ox}}{t_{ox}} (V_{gs} - V_{th}) \right]^{-1},$$

where μ is the mobility of electrons in the channel ($m^2/V \cdot s$), V_{gs} is the gate source voltage, V_{th} is the threshold voltage, and L W are related to the dimensions of the gate, length and width respectively. The equation models the resistance on the linear region. In the light of this analysis, this is conservative, since R_{tr} in saturation is lower than in the linear region and transistors operate in both regions, hence, the effective resistance becomes lower. This implies that the resistance of a long interconnecting wire becomes even more relevant, as will see in the delay section.

2.2.1 Scaling

The advances in the VLSI process technology have led to smaller dimension devices and at the same time, to larger available area for circuit integration. In order to understand how these advances will impact circuit design, a scaling theory was developed. A first-order MOS scaling model was introduced by Dennard et al. [DGY⁺74] and it is used throughout the literature [MC80] [WE85]. Basically it applies a dimensionless factor α ($\alpha > 1$) to all dimensions, voltages, and concentration densities. Table 2.1 presents the scaling effects. This particular table is from [WE85].

Device	Parameter	Scaling Factor (<i>alpha</i>)
	Length L_c	$1/\alpha$
	Width W_c	$1/\alpha$
	Gate oxide thickness t_{ox}	$1/\alpha$
	Supply voltage V_{dd}	$1/\alpha$
Result	Gate delay	$1/\alpha$
	Power dissipation	$1/\alpha^2$

Table 2.1: MOS Device Scaling

Based on this first-order scaling theory for MOS technology, smaller dimensions imply faster transistors, i.e., switching delay scales down by α , and the number of transistors on a same chip size scales up by α^2 . However, this only holds for small circuits. Performance of larger circuits is affected negatively by two factors: increased interconnection delay and increased interconnection area, the former using up area that could be used for logic.

Saraswat et al. [SM82] developed the MOS Scaling theory further to incorporate the effect of scaling on interconnection, classifying it in local interconnection and long distance interconnection. They take into consideration the scaling up of the usable chip area by a factor γ ($\gamma > 1$).

Parameter	Local Interconnection scale factor α	Long Distance Interconnection scale factor γ
Interconnection distance	$1/\alpha$	γ
Line resistance	α	$\alpha^2\gamma$
Line capacitance	$1/\alpha$	γ
Line response	1	$\alpha^2\gamma^2$

Table 2.2: Interconnection Scaling

The local interconnection scales down with α , though its RC product remains constant, meaning that the time delay does not scale down. The time delay problem is even more critical regarding long distance interconnection: signals have to travel longer distances because of the scaling down of the devices and also because of the scaling up of the chip size. Furthermore, the smaller devices handle less power, which limits their current driving capability. Hence charging/discharging the interconnection takes longer. These two scaling effects can be seen in Table 2.2. Figure 2.3 illustrates the interconnection scaling.

For instance, line resistance for local interconnection is $R = \frac{l}{w \cdot t}$ which implies in $R_\alpha = \frac{l/\alpha}{w/\alpha \cdot t/\alpha} = R \cdot \alpha$. For long distance, $R = \frac{l}{w \cdot t}$ which implies in $R_{\alpha\gamma} = \frac{l\gamma}{w/\alpha \cdot t/\alpha} = R \cdot \gamma \alpha^2$. A similar development applies for line capacitance. Line response becomes the RC product.

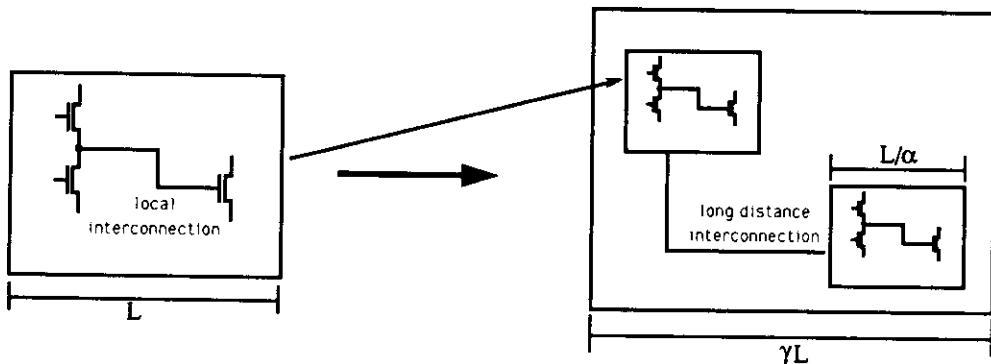


Figure 2.3: Interconnection Scaling

The first order scaling is a simplified model of the advances of the technology. It is reasonably accurate and quite useful to provide insights into issues that arise with the changes in technology. The physical limit for the MOS technology is around minimum feature size of $0.1 \mu\text{m}$. Table 2.3 [Spe88] shows the advances regarding feature size and area. The results were obtained by simulation and

extrapolation of the limiting factors in the manufacturing process.

Year	Minimum Feature Size μm	Area mm^2
1972	6	10
1975	4	15
1978	2.5	24
1981	1.5	40
1984	1	60
1987	0.6	100
1990	0.4	150
1993	0.25	240
1996	0.15	400
1999	0.1	600

Table 2.3: State of the Art Technology Evolution

2.3 Area and Interconnection

In VLSI, interconnection is known to be area intensive. Usually it takes up more than 50 percent of the total area. In [MRRS84] it is claimed that 60 to 80 percent of the chip area is taken for interconnection. Seitz in [Sei84] goes as far as claiming that interconnection takes usually 95 percent if taken at the transistor level, with 5 percent left to implement the transistors that effectively perform the computation. It is important to take this fact into account because it affects chip gate density and has two major consequences: less gates per chip because interconnection can be considered as an overhead in terms of area, and at the same time, larger distances between modules (functional blocks), affecting therefore, the performance since the increased distance implies increased delay.

Of course, interconnection cannot be avoided, since at the end it is what provides the communication internally and externally for the logic modules, which are the ones implementing the algorithms. This reinforces the notion that interconnection has to be designed carefully. The care is even more important for the upcoming technologies where interconnection will have a major impact on performance not only in area but also in time.

2.4 Delay

The effect of scaling in interconnection has been in focus in the literature over the past years. One of the most important results has been to show that interconnection delays will be of a major concern in the submicron technology arena [GMS87, NDN87, SPP87, BM85, SM82, MC80]. Scaling of interconnection per se was discussed in a previous section.

For the delay expression, we will use an approximate result by Sakurai [ST83]. This result is based on an exponential approximation. For the range of parameters we will be using, the relative error is claimed to be in the order of 1 percent.

Sakurai's result:

$$t = 1.02C_i \cdot R_i + 2.21(C_{tr} \cdot R_{tr} + C_{tr} \cdot R_i + R_{tr} \cdot C_i) \quad (2.3)$$

where C_i and R_i are the total capacitance and resistance of the interconnection, and C_{tr} and R_{tr} characterize the transistor. Figure 2.4 illustrate this.

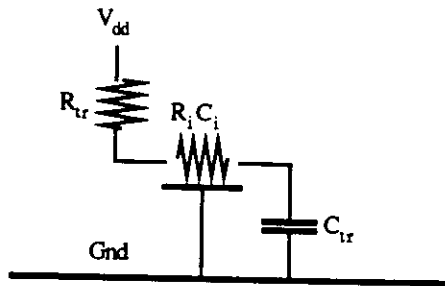


Figure 2.4: RC Model

2.4.1 Driving Schemes

By looking closer at equation 2.3 we may ascertain how delay behaves in terms of the interconnection length l .

$$t(l) \sim C_I R_I l^2 + R_{tr} C_{tr} + (R_I C_{tr} + R_{tr} C_I) l$$

We note that the delay can vary from a *constant* ($O(k)$), to *linear* ($O(l)$),

to *quadratic* ($O(l^2)$) behavior. We are considering metal interconnection where $R_{tr} \gg R_l$.

- $O(k)$: when $R_{tr}C_{tr}$ is the dominant factor. This is true for small l and for previous technologies where the interconnection did not play a significant role in delay (refer to Table 2.2).
- $O(l)$: for certain ranges of l and technologies when the dominant factor is $R_{tr}C_l l$
- $O(l^2)$: at submicron technologies, the dominant factor becomes $C_l R_l l^2$ as $R_{tr} \leq R_l$.

Based on this, it was found that the delay could be kept at $O(l)$ by breaking the line and buffering the signal (with so called repeaters) from space to space [GD85, WE85, Muk85, BM85]. By a similar reasoning in order to drive a high capacitance load, it should be driven by a series of drivers of incrementally larger size [HJ87, CPM86, GD85, BM85, Ram82, MR82].

With this in mind, we present and compare some different schemes for driving signals. We are only considering aluminum lines (metal wires) since we are interested in the lowest delay possible for long distance on chip. Other materials that are also used in interconnection like polysilicon and metal silicides have a considerably worse performance [SM82, Rei83] due to the high RC constant and are therefore used for very local interconnection. Similar study was done by Bakoglu [BM85].

The delay expressions for the driving schemes are based on expression 2.3 for each of the segments. We assume that the load on this network is equivalent to the gate of a minimum size transistor.

The driving schemes are (see Figure 2.5):

minimum size: A minimum size transistor driving the signal line.

$$t_{min} = R_i C_i + 2(R_{tr} C_{tr} + R_i C_{tr} + R_{tr} C_i)$$

segmented: In this case the line is segmented in equal size segments and each of them is driven by a minimum size transistor. These driver-segments are connected in series. The number of segments is such that it leads to a minimum delay regarding this scheme.

$$t_{seg} = \frac{R_i C_i}{k_s} + 2k_s \left(R_{tr} C_{tr} + \frac{R_i C_{tr}}{k_s} + \frac{R_{tr} C_i}{k_s} \right),$$

which is minimized by solving $\frac{\partial t_{seg}}{\partial k_s} = 0$

$$\text{hence, } k_s = \sqrt{\frac{R_i C_i}{2R_{tr} C_{tr}}}$$

capacitance driver: Since one of the major causes of the delay is the high capacitance of the lines, a scheme described in [MC80] is used. A series of drivers are cascaded in such a way that the next one is larger by a factor f , $f > 1$, therefore, more capable of driving current to charge/discharge capacitances. The number of drivers is optimal for the capacitance on the line. For ideal conditions, $f = e$ [MC80]. When intrinsic delays are taken into account, $3 < f < 5$ [HJ87].

$$t_{cap} = k_d f R_{tr} C_{tr} + R_i C_i + 2\left(\frac{R_{tr} C_{tr}}{k_d} + R_i C_{tr} + \frac{R_{tr} C_i}{k_d}\right)$$

$$\text{with } k_d = \ln\left(\frac{C_i + C_{tr}}{C_{tr}}\right) / \ln(f)$$

SC driver: (segmented capacitance driver) A combination of the segmented and the capacitive driver to achieve the minimum delay.

$$t_{SC} = k_s k_d f R_{tr} C_{tr} + \frac{R_i C_i}{k_s} + 2k_s \left(\frac{R_{tr} C_{tr}}{k_d} + \frac{R_i C_{tr}}{k_s} + \frac{R_{tr} C_i}{k_d k_s}\right)$$

2.4.2 Results

The results can be seen in Figure 2.6. The minimum size driver has the worst performance as expected. However, it is the one that consumes the least power and area.

The segment driver has a quasi-linear behavior: segments are added as distance grows when the delay of the line segment is comparable to the delay of the buffer. Otherwise, the delay would increase as the square of the line length because both, the resistance and the capacitance of the interconnection, increase linearly with the length, as it is the case with the min driver.

The capacitance driver has a significant improvement in the performance while the resistance of the interconnection is less than the driver resistance. From that point on, the RC of the line becomes the predominant factor and asymptotically this driver performs as the min driver.

The SC driver compensates for both, the capacitance and the resistance of the line. Segments are added as the resistance of the interconnection grows, and at

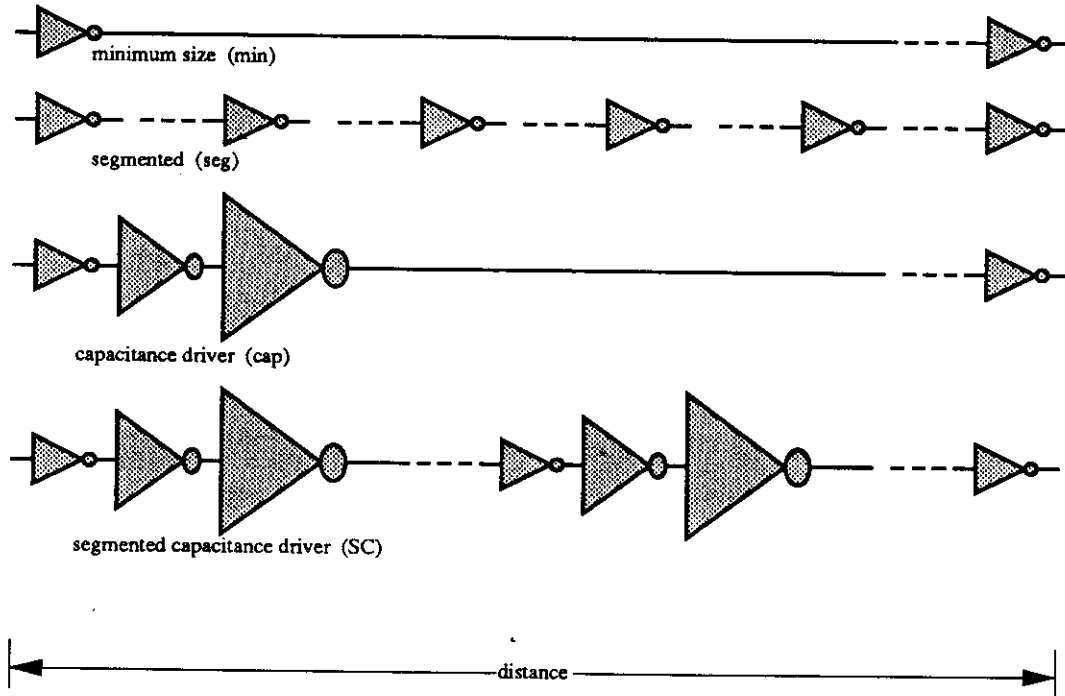


Figure 2.5: Driving Schemes

the same time, larger drivers are placed to adjust to the line capacitance. The values for k_d and k_s were found by numerical optimization. The split seen in the graph when compared to the capacitance driver occurs when the line starts to be segmented.

In Figure 2.6 we can observe an interesting fact regarding the relationship between the driving schemes and the technology stage: since for the current technology the split between t_{SC} and t_{cap} occurs beyond the diameter of the chip it means that the “bus solution” (so common in current designs) is a viable solution in terms of delay performance. However, for future technologies the results for low delay strongly suggest the use of segmentation, which is unidirectional by its nature, requiring a different approach to the chip communication problem. We also show in the figure, the accepted region for VLSI and WSI.

Furthermore, design decisions cannot only rest on these delay figures. In VLSI, equally important are the area and power needed for those different schemes when design decisions are being made.

An important point that hasn't been considered so far is the shape of the signal. After the signal passes through a long capacitive line, the transition from one voltage correspondent to a logical level to the other becomes slow due to

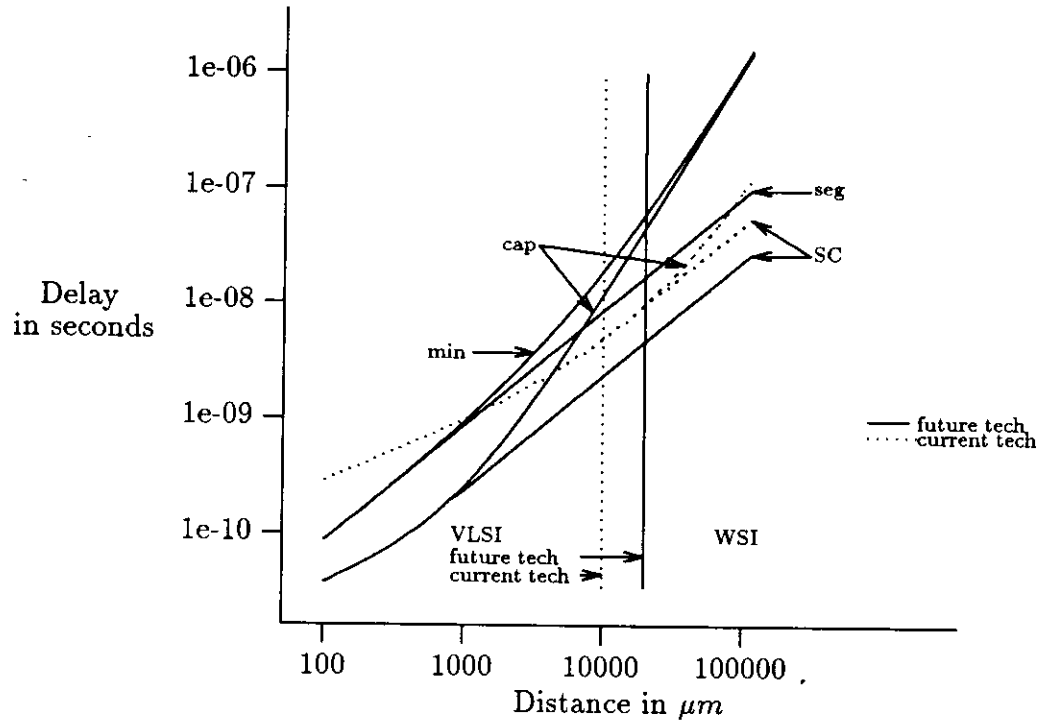


Figure 2.6: Delay versus Distance

low pass filtering characteristic of the line. These slow transitions increase the uncertainty of the logic level and more time is needed for the circuitry to decide, thus imposing more delay and possibly a metastable condition [MC80, GD85]. In this condition the level of the signal is such that it cannot reliably be interpreted as either high or low and, theoretically, it may remain in this state indefinitely. The regeneration of the signal helps to overcome this additional delay, as it is the case on the segmented lines where drivers are strategically placed. A piecewise analysis method which can be used to predict the logic propagation delays taking into account the finite slope of the input was developed by Bayruns [BJJF84].

2.4.3 Delay and Throughput

In previous works, the emphasis has been on the time delay that signals incur as they travel long distances in the chip [BM85, BM84, SM82, MR82, Ram82]. However, as important as it is to lower the delay, it is to increase the throughput, since both factors affect the transfer of information.

Therefore, it is not just a matter of how long it takes to send information from one point to another, but how much information can be sent at a given time.

One of our conclusions in the last section was that dividing up the long distance

interconnection wire and buffering the signal at those points, effectively segmenting the wire, showed time behavior improvements. As a side effect, this also created “zones” of information, each zone corresponding to a segment. Because of the buffering, the segments become logically isolated, which implies that all zones may be carrying different information at a given time. As exemplified in Figure 2.7, the packets of information move from one zone to the next, from source to destination like in a pipeline. This is the property that we want to exploit in order to improve the throughput. We should also note, however, that the segmented interconnection becomes unidirectional since buffers have been inserted (as opposed to a continuous wire which electrically “broadcasts” the signals)

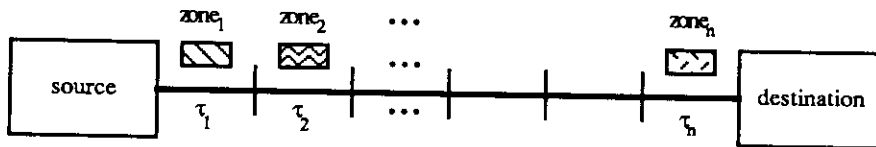


Figure 2.7: Zoning of a Interconnection Wire

In terms of throughput, the maximum throughput is the inverse of the highest segment delay. Trivially, one finds that the maximum throughput for a given distance, is for segments with equal delay, i.e., equally spaced buffers assuming same electrical characteristic for all segments and load capacitance at the end of the wire equivalent to the buffers.

An interesting compromise can be accomplished between the end to end delay and the information throughput of an interconnection line. We define a benefit function G ,

$$G(s, l) = \frac{Tp(s, l)/Tp(1, l)}{Dl(s, l)/Dl(1, l)}$$

where $Tp(1, l)$ is the throughput over the interconnection with length l and with a single driver at the source. $Tp(s, l)$ is the throughput for the interconnection with s segments. $Dl(1, l)$ and $Dl(s, l)$ are the analogous for end to end delay. Of course $Tp(s, l)$ takes into consideration the pipelining effect.

$G(s, l)$ is the proportional to throughput and inversely proportional to delay. Hence, the benefit is high when the end to end delay is small and the throughput is high. Optimizing G will lead to a good operating point of the system, e.g., a good compromise between delay and throughput. A concept analogous to benefit function G , known as “Power” [Gai83] was developed for computer networks. Power is defined as the ratio of Throughput over Delay. Again, power optimization leads to an effective operating point of a communication link.

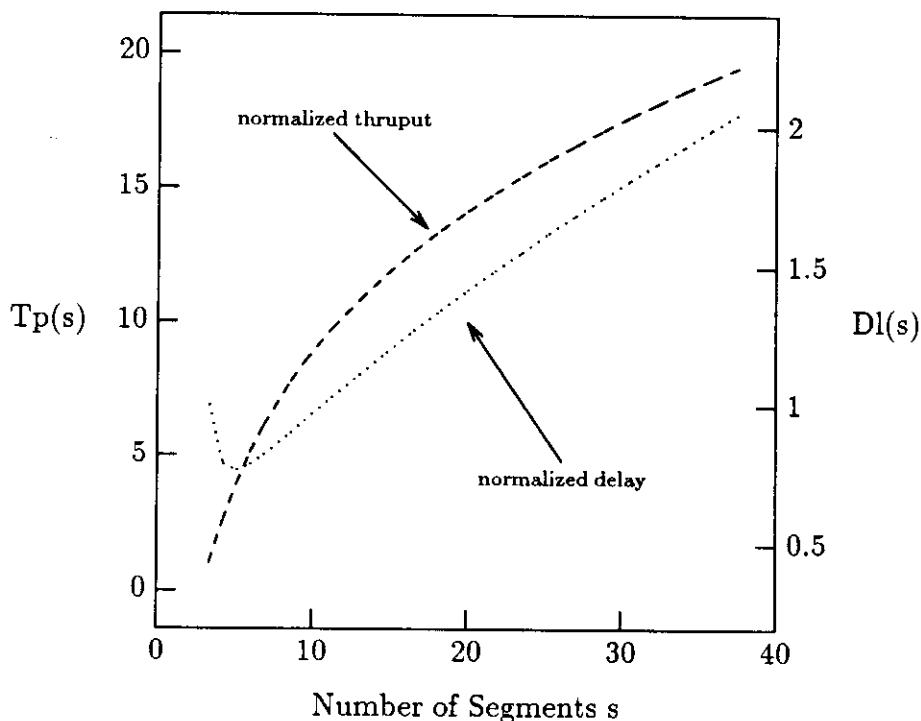


Figure 2.8: Throughput and Delay versus Number of Segments

The graph in Figure 2.8, shows the normalized throughput ($Tp(s, l)/Tp(1, l)$) and the normalized delay ($Dl(s, l)/Dl(1, l)$) for varying number of segments s and for an interconnection with length l , where l is the typical VLSI diameter. The delay values were derived from the *SC* driver expression. The throughput is the inverse of the delay of a segment. As the number of segments increases, the delay decreases until the number of segments approaches the optimal number of buffers (as seen in the previous section). From that point on, the delay increases because of the added delay of the buffers. The graph in Figure 2.9 shows the functions G versus s . We note that for a small number of segments, the benefit function G is low because the delay is high (less segments than the optimal number) and the throughput is low (small number of zones). As we increase the number of segments, we cross the point where the delay is minimum. If we continue to increase the number of segments, we will increase the throughput and the delay. This means that after some point, the benefit of the increased throughput will be offset by the increased delay. The reason for the increased delay is that for each added segment, the buffer delay accumulates to the total delay.

We observe that the sensitivity of G is high for a small number of segments. This implies that there is a high benefit even with a small number of segments. In terms of VLSI, this is interesting since more buffers mean more power dissipation and more occupied area and that a significant gain can be made with a small number of segments.

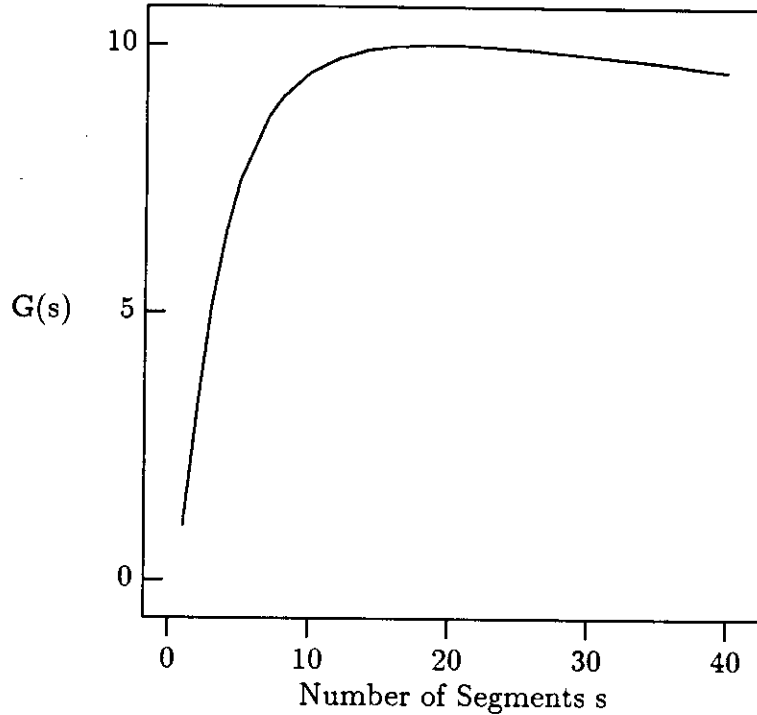


Figure 2.9: G versus Number of Segments

We should remark that G_{MAX} is not the absolute optimum for all applications, and that we should use the values for G as a guideline for the operating point regarding the compromise between throughput and delay.

2.5 Conclusions

In this chapter we have shown some of the trade-offs that will have to be considered when designing a long distance interconnection.

Moreover, the results obtained here strongly suggest that a communication solution for the upcoming technologies should be based on a slotted communication architecture so that low delay and high throughput can be attained. We envision the communication being broken down into different levels, each one with different characteristics. For example, three levels, the lowest being the point to point, the middle level using bus and the higher level utilizing some kind of network on chip.

Most of the VLSI chips built so far at the highest level have some kind of shared interconnection, mostly bus. This can be noticed in general purpose microprocessors, memories, dedicated processors, and many other designs.

We conclude by claiming that the interconnection of a VLSI chip should also be designed hierarchically and that at the uppermost layer, there should be a network that benefits from the segmentation of long lines and also profits from the pipeline effect introduced by the segmentation.

Chapter 3

Network Choices for VLSI

3.1 Introduction

The communication among the elements of a multiprocessing or multimodule system is the key to the exploitation of parallelism and performance. Interconnection networks is a wide field and a rich literature is available, presenting many different topologies, architectures, and implementations. It is not our aim to go into detail of these interconnection networks. The purpose of this chapter is to discuss topologies for VLSI adequate for the advancing technologies in the light of parameters presented in the previous chapter (Chapter 2), namely, delay, throughput, and area.

We first present a taxonomy for interconnection in order to have a framework for our discussion and will later complement it with issues that we consider important for VLSI. Further on, we will discuss different topologies under this framework.

3.2 Taxonomy

The taxonomy presented here is based on Feng's survey of networks [yF81]. His motivation was the essential points of a cost effective network from a practical point of view. We will use this as a starting point for our discussion on the intrachip VLSI network.

In selecting the architecture of an interconnection network, four design decisions are identified. They concern the operation mode, the control strategy, the switching method, and the network topology.

Operation mode: Communication is identified as *synchronous* for processing in which communication paths are established synchronously; and *asynchronously* in which connection requests are issued dynamically. Systems with both operations mode are also possible. Synchronous mode is used most in broadcast systems or small systems for data and/or instructions. Asynchronous communication is more suited for multiprocessing in which connection requests are issued dynamically.

Control strategy: Interconnection networks are typically made up of switching elements and their links. The interconnection functions are performed by properly setting the control of the switching element. This functions can be managed by a *centralized controller* - centralized control strategy - or by the individual switching elements in that case a *distributed control* strategy. This falls in the discussion of central or distributed control. Centralized control is usually simpler, however, failure and congestion are the negative side to it. On the other hand, distributed control is more complex in its design since issues like liveness, fairness have to be implemented.

Switching methodology: The two major switching methodologies are *circuit switching* and *packet switching*. In circuit switching, a physical path is actually established between a source and a destination. In packet switching, data is put into a packet and is routed throughout the network without establishing a physical connection path. In general, circuit switching is more suitable for bulk data transmission (large granularity), and packet switching is more efficient for short data messages (fine granularity) because of the set up time overhead compared to the address information in the packet. Combining the capabilities of both, circuit switching and packet switching, we identify integrated switching.

Network topology: A network topology can be represented by a graph where edges and nodes represent respectively the communication links and the switching points. Most of the topologies are regular and can be grouped in two categories: *static* and *dynamic*. The classification is depicted in Figure 3.1. For instance, ring is a regular static one-dimensional topology: links between two processors are passive and cannot be reconfigured to direct the connection to other processors. Dynamic topologies allow link reconfiguration by setting the network's active switching elements. A different way of looking at these categories is to verify how the interconnection itself is designed. If there is only one interconnecting layer between two nodes the topology is static and cannot be reconfigured (like a bus topology); on the other hand, if the interconnection layer consists of more than one layer, it can be reconfigured (like shuffle-exchange).

- The *static* topology category can be classified according to the number of dimensions in the connection space. Bus, linear array, are examples of one-dimensional static topology. Ring, star, tree, near-neighbor mesh, and

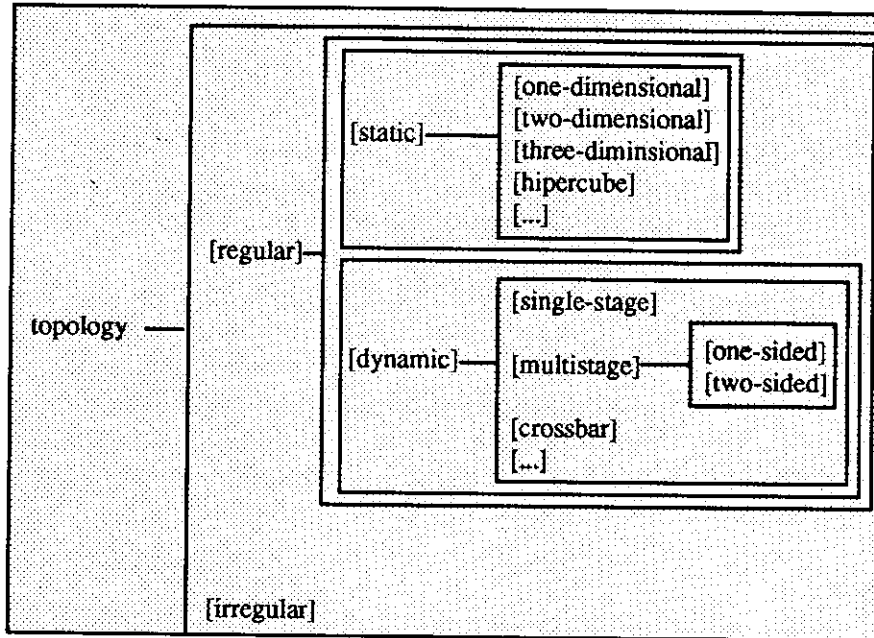


Figure 3.1: Topologies Classification

stolic arrays are two-dimensional. Three-dimensional topologies include chordal ring, 3-cube, and 3-cube-connected-cycle [PV81]. Furthermore, a d -dimensional w -wide hypercube contains w nodes in each dimension, and there is a connection to a node in each dimension.

- The *dynamic* topology has three different classes: single stage, multi-stage, and crossbar. Single stage network is composed of a stage of switching elements cascaded to a link connection pattern. The shuffle-exchange network is a single-stage network based on a perfect-shuffle connection cascaded to a stage of switching elements. Multi-stage network consists of more than one stage of switching elements and is usually capable of connecting an arbitrary input terminal to an arbitrary output terminal like benes, baseline, delta, omega, banyan networks [yF81, HLSM82, Fra81, CLYP81].

3.3 Performance Criteria

In order to select the appropriate design decisions presented in the previous section, we have to define a set of performance criteria and design parameters under which the network should operate.

Some common performance criteria are: delay, throughput, fairness, and cost. Delay is a measure of the time packets take to travel from origin to destination.

It may relate to the average time or to the maximum time. Throughput measures the number of packets that go across the network per unit time. Fairness refers to the equality of distribution of the bandwidth. Cost is related to the complexity and to the physical size of the design which determines its yield.

These criteria are not orthogonal, therefore, they conflict with one another. For instance, high throughput (desirable) may lead to high delay (undesirable) and low fairness (undesirable); low cost (desirable) by using serial communication may lead to high delay (undesirable).

Some of this conflicting issues may be assisted by strategies in order to meet the performance objectives:

- *Routing mechanism*: How packets are routed from source to destination. The mechanism helps to alleviate congestion by deviating packets to less used paths.
- *Access protocol and flow-control*: How packets get access to the network and how congestion is handled by slowing down the traffic coming to the network. By slowing down the traffic, delay decreases and fairness may improve.
- *Redundancy*: How redundancy and reconfiguration mechanisms may reduce manufacturing cost by increasing the yield or decreasing the maintenance cost by prolonging the useful life of the system.

3.4 Design Criteria

As important as the performance criteria for the network, are the design criteria (i.e. design variables and constraints) peculiar to VLSI design, which may determine how the performance criteria may be attained. Some of these issues are:

- *Layout*: Deals with how networks may be laid out on chip. There are a series of constraints since the basic space is a two dimensional one with a very limited number of independent layers that may overlap. The common value for these different layers is two or three. Some other important characteristics of this mapping are regularity, distance between switches, and the capacitance involved with this mapping. Care should be exercised to avoid complex layouts with wires crisscrossing; this would make it more prone to noise, crosstalk, greater capacitance, and also make it more difficult for power distribution wires (power grid).

- *Number of ports per switch:* A large number of ports implies a more complex switch that needs more circuitry and consequently more area. This usually leads to slower switches with higher delay. Furthermore, more power would be needed for the extra load (logic and drivers), and an increased difficulty in the mapping, discussed above. Similar conclusion was reached by Fujimoto [Fuj83].
- *Number of ports per bus:* If a multiaccess, multipoint bus is used, a larger number of ports involves higher capacitance on the bus since ports are capacitive loads. Hence, high driving currents are needed for faster transmission, implying larger drivers and larger dissipated power. The end result is an increase in delay through both, the driver chain and bus.
- *Height and width of wires:* Height and width of wires are parameters of the wire capacitance and resistance (see Chapter 2) and consequently parameters for the delay. However, the delay cannot go below some value because it is bounded by the maximum sectional current that may flow across metal lines. Currents above the limit produce metal migration [MC80, GMS87], which will literally carry metal away from the "hot spots" and eventually break the connection. Using wider wires will increase the capacitance, which in turn will need more power (current). The height of the wire is limited by the technology. The "solution" lies in careful design.

3.5 Critique of some Topologies for VLSI

As we can see there is a wide range of interconnection network topologies, (see Figure 3.1), from the bus network at the lower bound to the crossbar at the upper bound of regular topologies. Neither are appropriate for highly parallel arrays [HLSM82]. The bus network is architecturally simple, but in an n -processor network only $1/n$ of the bandwidth is available for each processor. The crossbar network with every processor connected to all others, uses n^2 switches to connect n processors. For VLSI, this structure becomes unfeasible because of the number of switches required and the layout complexity of interconnecting wires. Below we will discuss some of the most commonly proposed topologies for future VLSI technologies.

- *Bus:* Very regular and simple, however, there is a high penalty in delay for long distance and for the number of ports connected to it so that the transfer rates become quite limited. Also, since it is a unique shared resource, contention becomes high. This is the most commonly used intra-chip communication in current VLSI designs, though it will not be appropriate for

future technologies. However, a hierarchy of busses could be an appropriate solution for traffic patterns that hold physical locality.

- *Ring*: Because of the slotted nature of the optimal driving scheme for long distance interconnection (detailed in Chapter 2), ring offers a simple solution. With the use of an appropriate protocol, it is possible to have more than one active slot (a frame carrying data) at a time, and so a higher throughput may be achieved; however, delay is high ($n/2$). The layout can be easily accommodated for VLSI.
- *Mesh*: A regular mesh, with the placement of switches based on the optimal long distance driving scheme. A mesh-connected network is exemplified in [TH88].
- *Trees*: May have a good layout, the so called *H - tree* [MA71, MR79], but non uniform distances may become a problem as well as routing, leading to low throughput and high delays and eventually to bottlenecks at the root (or close to the root). This topology is very sensitive to application.
- *CCC*: (Cube Connected Cycles) is an interesting arrangement, with high regularity and small number of ports. However, it does have non uniform interconnection length. From [PV81], length can be as high as half the chip size in the standard layout, to full chip size in the more economical layout. Hence, in spite of its nice placement, the delay through these long interconnections can greatly upset the performance of some algorithms.
- *Hypercube* (binary-cube): In a planar mapping of this topology (for a binary 6-cube see [Sei85]), the length of the links and the number of ports increases as dimensions grow. The layout is complex and costly in terms of area. Although routing seems simple, it can become quite hard when an optimal routing and flow control solution is sought.
- *Omega nets and equivalents*: Given that the delay is a function of the link length, the problem becomes the layout of the switches and the links among them. It is not possible to distribute the switches on a planar layout and connect them with equal link length (manhattan distance). Hence, this does not seem to be an appropriate solution for VLSI interconnection.

3.5.1 Discussion of Solutions

The topologies presented, from the link point of view, fall in two categories: 1) point to point, 2) multiaccess.

The point to point fits very well with the results obtained on Chapter 2. At each point signals are amplified, the distances between the points can be such as to minimize delay, and the *zoning* allows for the pipeline effect, therefore higher throughput may be achieved.

The multiaccess should be a hierarchy of small busses so that the lengths of the busses are small and load (number of ports connected to it) is also small. By the virtue of the hierarchy, locality can be exploited by the applications which also improves the throughput. This ought to happen in a well designed system. (discussion in Chapter 1 - distribution of clock signals)

Links can be further classified as bidirectional or unidirectional. In multiaccess, busses are by their nature bidirectional, while in point to point, if we take into consideration the results of Chapter 2 (driving schemes), the suitable solution calls for unidirectional links.

In a mesh topology, switches are connected by point to point links which can be unidirectional or bidirectional. The major difference in terms of routing between uni and bidirectional, is that path lengths in the bidirectional case are, in most cases, lower.

In terms of implementation, in the bidirectional case, there are two possibilities: a single physical link that switches the direction of flow following some protocol (half-duplex) and a dual physical link, each link dedicated to one direction (full-duplex).

We will not consider the full-duplex solution because it is too expensive in terms of area for additional circuits and wires, besides the added complexity of the control circuit itself.

To compare the unidirectional and the half-duplex scheme, we refer to Figure 3.2 and Table 3.1. We assume that the system has a regular topology (mesh) for reasons discussed earlier and that it is slotted with time τ , which is also the propagation plus circuit delay between adjacent switches. For the unidirectional case, links alternate direction in both orientations, vertical and horizontal. For the half-duplex, t_{sw} is the time it takes to switch direction. In Table 3.1, values in brackets are the worst case situations.

The last line of the table shows the average time under a uniform destination assumption. We notice that the lesser distance of the half-duplex scheme does not show a major delay reduction over the unidirectional scheme. Furthermore, if $t_{sw} \geq \tau/4$, the unidirectional scheme comes out the winner. One also finds that, as distance on the grid between source and destination increases, the difference in manhattan distance between the two schemes decreases.

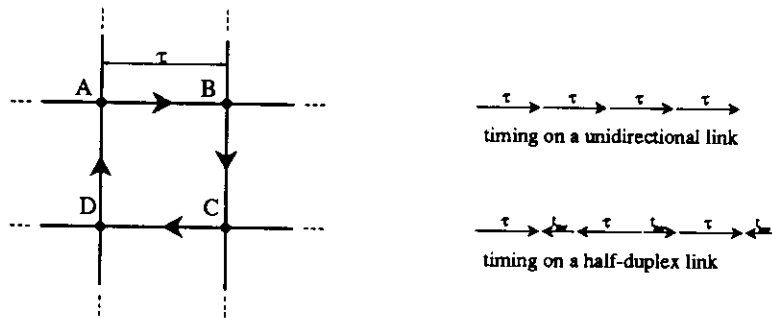


Figure 3.2: Part of the Network Mesh and Timing

from \rightarrow to	unidirectional		half-duplex	
	distance	time	distance	time
A \rightarrow B	1	τ [2τ]	1	τ [$3\tau + 2t_{sw}$]
A \rightarrow C	2	2τ [4τ]	2	2τ [$6\tau + 4t_{sw}$]
A \rightarrow D	3	3τ [6τ]	1	τ [$3\tau + 2t_{sw}$]
average	3τ		$8/3\tau + 4/3t_{sw}$	

Table 3.1: Time Comparison Between Unidirectional and Half-Duplex Links

In conclusion, we believe that the additional complexity of bidirectional links (half or full duplex) is too high a price to pay for the marginally shorter number of hops between node pairs, when compared to unidirectional links. Consequently, we will base our solution on the unidirectional scheme, which is simpler, needs less area and power and operates faster on a link basis.

3.6 “Appropriate” Networks for VLSI

Following Feng’s design criteria, the space of possible network solutions is given by the cross product of the set of categories for each design decision. Applying the VLSI constraints discussed earlier, we obtain a subspace of solutions that we will explore further.

- *Operation mode:* should be asynchronous because of clock skew and clock distribution problems in VLSI [FK85] and delay issues discussed in Chapter 2.

- *Control strategy:* should be distributed based on the same reasons. A timely exchange of information is not possible.
- *Switching methodology:* for most applications packet switching seems more appropriate than circuit switching due to the nature of traffic, usually short messages. The problem with circuit switching would be the overhead of setting up the connections.
- *Network topologies:* based on the earlier discussion, mesh and ring seem the most appropriate for VLSI topology.

In order to come up with an appropriate and cost-effective interconnection switching network for a class of applications, compromises among basic characteristics must be reached.

Chapter 4

VLSI Networking Approach

4.1 Introduction

In this chapter, we discuss the network approach to deal with interconnection at the highest level of the VLSI chips of future technologies.

We start by showing the hierarchy of interconnection at the present technology and how *scaling* affects it. We also present the advantages of the networking approach, the difference between the proposed scheme and other networks.

4.2 Motivation

The hierarchical approach to VLSI design is a must due to the complexity involved [MC80, BK83, UKH85]; specifically, transistors are *bundled* into gates that in turn create functional blocks, which in turn are *bundled* into modules and so forth. At each increasing level on the hierarchy, more functionality is added. This is easily observed in practically all VLSI designs: microprocessors, memories, memory management units, floating point units, and many others.

The *bundling* is accomplished by the interconnection whose function is to transport signals from one transistor to another, from one gate to another, and so forth. The hierarchy that we see applied to the functional aspect, also applies to the interconnection, where each level has its own requirements. This is illustrated in Figure 4.1. Mead in [MR82] suggests that the number of wires must decrease exponentially as a function of their length, also implying that efficient designs must have a hierarchical structure.

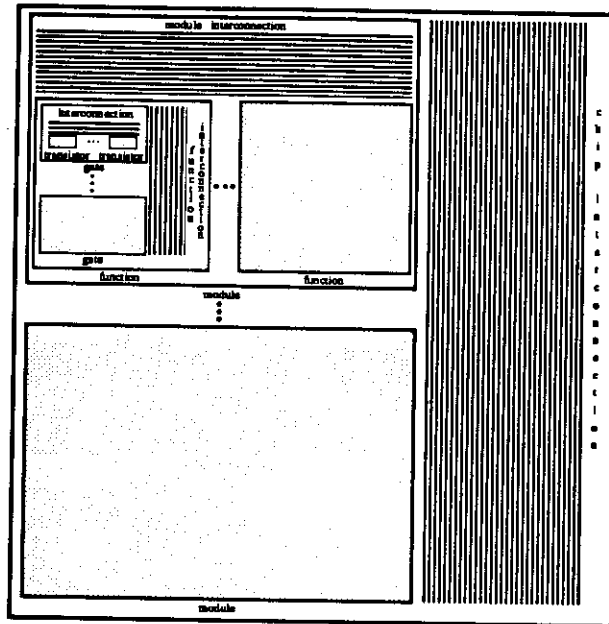


Figure 4.1: The Conceptual Hierarchy of VLSI Chip

In the current technology, the highest level of the hierarchy of intrachip interconnection is the interconnection of modules. The next level is outside the boundary of the integrated circuit. The module interconnection has so far been handled, in most cases, by point to point interconnection and by strategies based on bus. The bus communication strategies are very common and rely on a central model with tight control.

Given the technological advances presented in Chapter 2, we see the minimum feature size being scaled down (factor $1/\alpha$, $\alpha > 1$), and chip size being scaled up (factor $\gamma > 1$). Gate density is, hence increased and so is the available chip area. In this context, modules from the current technology, which were at the highest level of the on chip hierarchy will be *bundled* into module-clusters, and these clusters will become the next and last level on the on chip hierarchy. Figure 4.2 shows this.

To provide communication between these module-clusters we are proposing a networking scheme. We see networking on chip as a necessary step to keep up with the advances in technology. One of the motivations for the network framework is based on the delay results obtained on Chapter 2. In order to cope with the increasing line capacitance and resistance for long distance, and hence increasing delay, we observed that solutions should be based on reducing the length of the

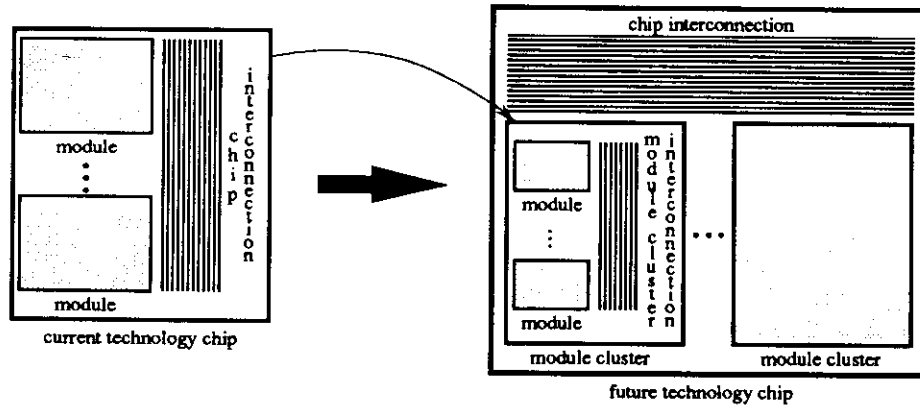


Figure 4.2: The Scaling of the Hierarchy of a VLSI Chip

lines. Signals traveling long distances should have active elements properly placed on the interconnecting wire. Two benefits of this solution are: 1) delay does not grow as dramatically, and 2) the *zoning* effect that allows the coexistence of different information along the interconnection (pipelining). We will take advantage of both effects in our networking scheme.

One reason why networking has not yet happened in the current technology is because the interconnection delay problem presented in Chapter 2 is most significant for the developing technologies, that are still in experimental phase. The gain for today's technology, if any, would be small because of the tradeoff between the area for modules and the area for the intra-chip network. This will change, however, as gate density as well as chip area increases.

4.3 Networking

A strong point of networking is the development of an interface between the intercommunicating blocks. This interface creates the separation of the communication functions such as routing and flow control from the blocks themselves. There are many advantages for such a separation:

- better control over the complexity of design, since hierarchy is enforced,
- standard communication interface,
- communication decisions done on the network, relieving blocks from such functions,

- savings on area for interconnection (due to sharing communication channels),
- flexible addressing,
- combination of blocks that operate at different speeds.

The size of such a network is quite limited and is likely to vary from 16 to 64 nodes. Many tradeoff decisions have to be made to find out an appropriate number. Given that everything has to be integrated on a certain area, a balance between number of modules and their sizes has to be achieved. If the number of modules becomes too high, the overhead of the network in terms of area (for circuits and wires) and power needs become high and modules have a lower functional integration (they are smaller). In the opposite case, it might happen that modules become big and sluggish because of internal delays and network resources inadequate to support efficient communications. This will affect negatively the overall performance of the chip.

It is also important to point out that a network architecture providing full interconnection (as the one we propose) is not the ideal solution for all intra-chip communication situations. For specific architectures that can be categorized as systolic arrays [Kun82] other communication schemes will perform better. In that case communications usually occur only between neighbors, and there is no need for an effective and high performance global communication scheme, which is what we are aiming at.

4.4 Differences from Conventional Environments

The networks that we propose here are very different from conventional networks such as LANs or WANs. The goal of our networks is to achieve global high performance on intra-chip VLSI communication. Therefore, a very specialized set of requirements (different from conventional networks) must be taken into account. We envision these networks as characterized by:

- assured delivery,
- streamlined protocols,
- fast routing decisions,
- small packet size (one bit word wide),
- regular topology matching the two dimensions of integrated circuits.

To achieve the assured delivery we are assuming a perfect channel model. Once the subnetwork has accepted a packet, it can be assumed that it will arrive to its destination within a bounded time (no loss, no buffer overflow). Protocols should ensure that there is no buffer overflow. Loss due to failures is very unlikely, with probability smaller than the probability of failure of a module. (Recall that faults are a function of used area and switches are significantly smaller than modules [SAS83, KB84]). The motivation for such assumption is that it simplifies the design of the switches, making it feasible to implement them at this level of the hierarchy.

The protocols have to be simple and efficient since, in order to achieve the high performance, the logic decisions should be processed in the order of a few gate delays. Here the previous assumption of perfect channel comes very handy: no need for acknowledgement, time-out, and retransmission mechanisms. If those mechanisms were necessary, great amount of area would have to be dedicated. Besides, they would lower the response time of the protocol in two folds: circuits would be more complex and slower, and the protocol itself would have to dedicate bandwidth and processing time to handle them. Windowing mechanisms, common in other networks, are also prohibitively expensive in this domain. Obviously, there should be error checking of the transmissions. However, this should be at a higher level, such as process level, for example.

Routing of packets from source to destination, based on the same assumptions as the access protocol, has to be simple and efficient. The time frame magnitude of the packets does not allow for any complex routing computation.

The size of the packets should be small because at this level on the hierarchy the granularity of information is small. Large packets would mean higher use of the communication resource in terms of time and space (contention and buffer space). In our study we have chosen packets that carry one word of information (32bits) in parallel. This implies that links have 32 lines of data plus address and control lines.

The topology issue is also very important: we should rely on topologies that fit nicely within the delay constraints presented in Chapter 2 and the planar characteristic of VLSI. As we saw earlier, node positioning should be at distances related to the need of amplification/regeneration in order to attain the minimum delay.

In the LAN arena, ring networks are usually criticized due to reliability problems because of the active elements (switches) [FT84]. In the case of the integrated network these problems are not so critical since, as mentioned earlier, the area for the switches is small in relationship to the total chip area and faults are a function of area.

Chapter 5

The Grid/RingNet Solution Approach

This chapter begins with our first attempt to develop a network on chip satisfying the VLSI environment. It is called GridNet for reasons which will become later obvious.

We start by describing the basic topology, components, and protocol, followed by results and analysis. The shortcomings of this approach are also presented.

We then introduce several improvements upon this basic scheme, focusing mainly on protocols and topologies, which are detailed in the following sections.

5.1 GridNet

GridNet (shown in Figure 5.1) incorporates the features discussed in previous chapters, such as regularity, distance, simplicity, and use of unidirectional links

5.1.1 Topology

The GridNet topology is a regular grid (mesh) and it is composed of blocks (a generic VLSI module), switches, links, and frame generators. At the intersection points of the mesh, e.g., where rows and columns intersect, are the switches (i.e. the nodes of the network). Blocks or modules (the building blocks at that level of the hierarchy) interface with the network through the switches. We assume there is one block per switch.

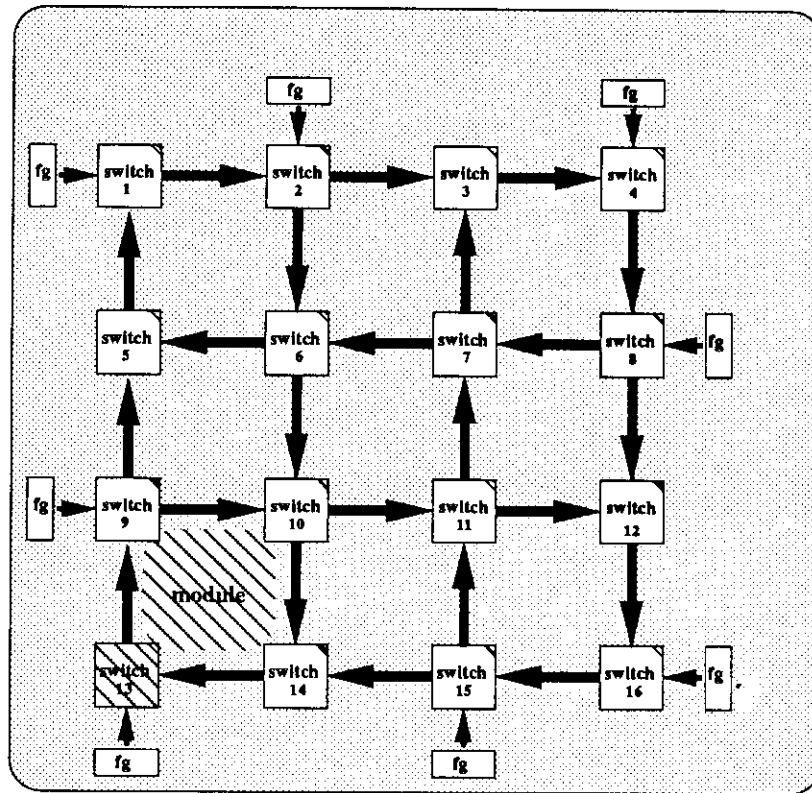


Figure 5.1: GridNet

Packets travel from switch to switch via links which are unidirectional, e.g., $switch_2$ in Figure 5.1 can send directly (one hop) to $switch_3$, but a packet from $switch_3$ to $switch_2$ must loop around and cover several hops. The directions of the rows and columns alternate between right to left and left to right, and between bottom to top and top to bottom respectively.

5.1.2 Protocol

At the beginning of each row or column there is a frame generator that generates empty slots (frames, containers) addressed to the individual switches down the line (i.e. polling mode). This provides the control information needed by the switches to put packets in the links. The empty slots are generated with addresses that cover all the switches that can send (the last one is not in the polling cycle) on that particular line. The rate at which frames are generated is $1/delay_{fg}$, where $delay_{fg}$ accounts for the propagation delay between switches plus the time taken by a switch to process a packet.

Figure 5.2 shows the timing of frames on a line (row or column). The shaded

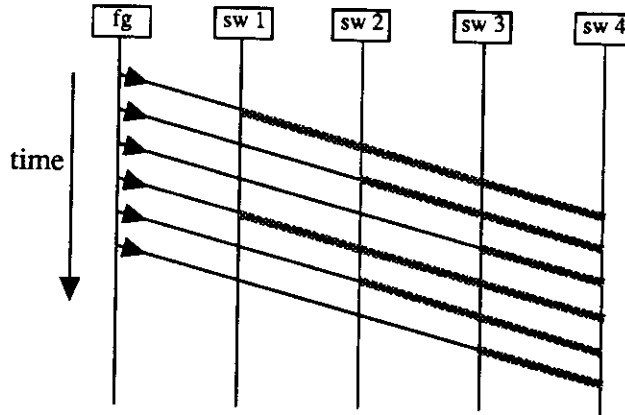


Figure 5.2: Timing of Frames

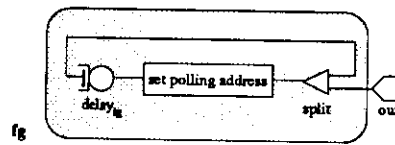


Figure 5.3: Frame Generator

area indicates the *possible* occupation of a frame by the switch where the shade starts. It shows the polling cycle that scans the set of switches $\{1, \dots, n - 1\}$ in order and after sw_{n-1} it goes back to sw_1 . The frame generator is shown in Figure 5.3.

Switches (Figure 5.4) are placed in the crosspoints of the grid and are composed of two line handlers (Figure 5.5), one for each direction. At the line handlers, the routing of packets is decided by the use of a predefined routing table (RT), e.g., routing is fixed and non adaptive. The RT is addressed by the packet destination and has three possible outcomes: 1) *arrival*, when the destination is equal to the switch address; 2) *change direction*, when the packet is to be sent out to the other line (switched traffic); 3) *moveahead*, when the packet is moved on along the same line to the next switch (through traffic).

Packets can be inserted in frames at polling time. The ones already in the network (the ones that have changed direction) have priority over new packets (the ones generated at the local node). Due to the asynchronous nature of frames and independence of the polling sequence, there is a need for buffering at the switches.

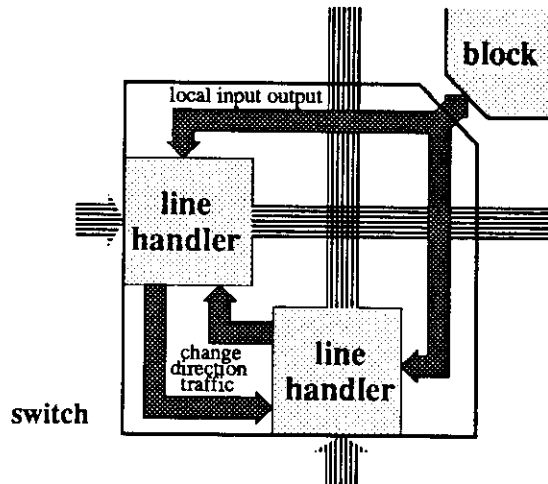


Figure 5.4: The Switch

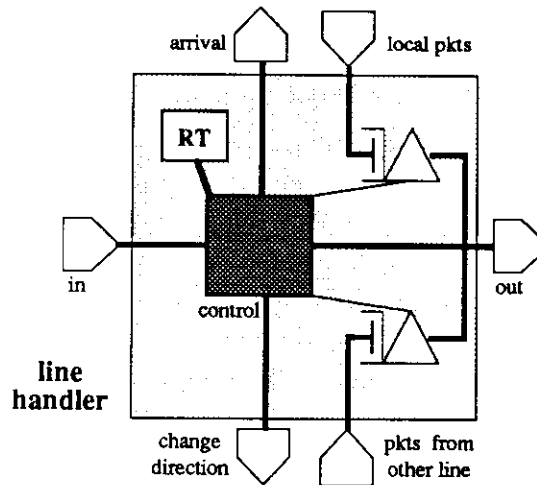


Figure 5.5: The Line Handler

The protocol can be described as:

```

at frame tick do (* generated by the frame generator *)
  case of RT(packet_destination)
    arrival:
      buffer packet in arrival buffer
      mark frame empty
    change_direction:
      buffer packet in other_node_switched_queue
      mark frame empty
    else :
      send frame with current packet
  
```

```

                                (* the one that came on this line *)
    at polling do
        if switched_queue not empty then
            send head_of_the_queue
        else if local packet waiting then
            send local packet
        fi
    od
od

```

5.1.3 Results - GridNet

A 4 by 4 network was simulated using RESQ (Research Queueing Package) [SBL+86] with the following assumptions:

- $delay_{fg} = 1$, and is used as the time base,
- uniform destination. All nodes send packets to all, excluding themselves with a uniform distribution,
- exponential interarrival time of new packets (local packets) with waiting room equal to one. The limit of one buffer was necessary to limit the number of states of the simulation; however, it also makes sense in a real system when hand-shake is used (i.e. a module can present a new packet after the previous one has been accepted by the network). In our study, no arrivals are accepted until the last arrival has been put on a frame.

The results of the simulation are presented in Figure 5.6 through Figure 5.9. The graph in Figure 5.6 shows the average delay taken by packets from the time they are generated till they reach their destination (in frame units) as a function of packet interarrival time.

Figure 5.7 shows the average aggregated throughput of the network (in number of packets per frame unit) and graph in Figure 5.8 shows the average throughput per module as well as the maximum and minimum throughput that they have achieved. The standard deviation of the modules throughput is also presented.

The graph in Figure 5.9 shows the average aggregated throughput on the links combined with the standard deviation and the maximum and minimum throughput on them.

Comparing with theoretical limits shown on Figure 5.6 and 5.7 as dotted lines (lower limit for time and upper limit for throughput), the results were not very

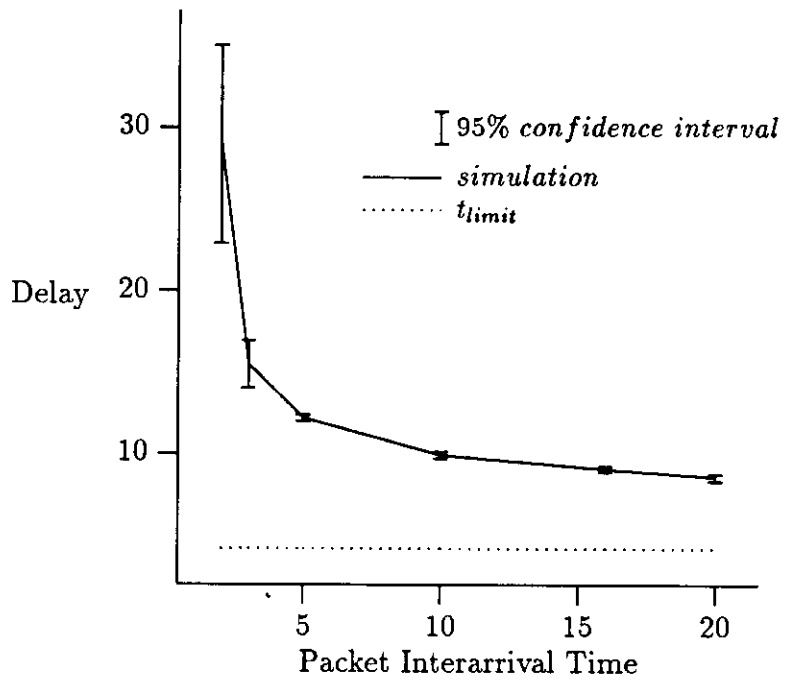


Figure 5.6: Delay versus Arrival Time for GridNet

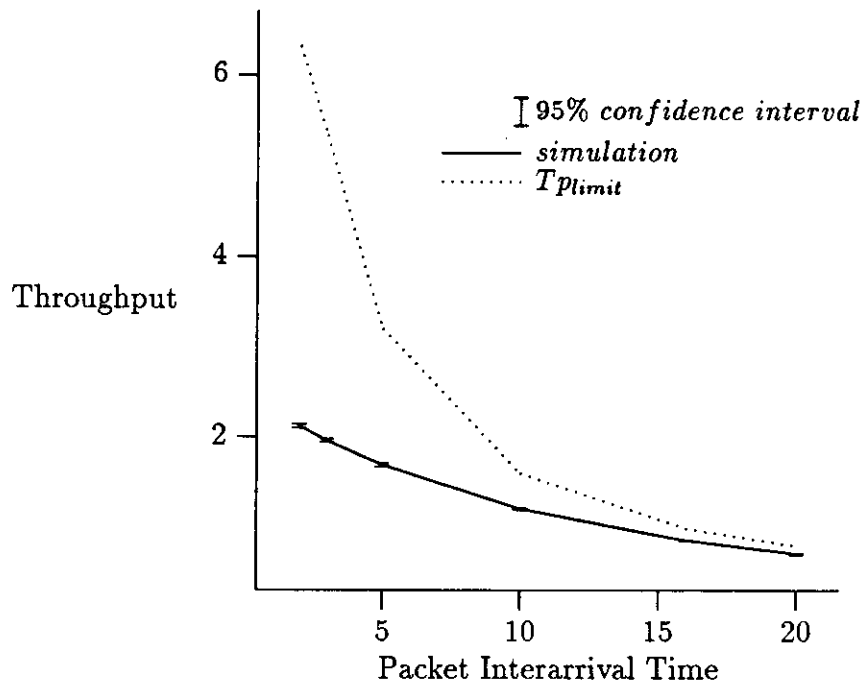


Figure 5.7: Throughput versus Interarrival Time for GridNet

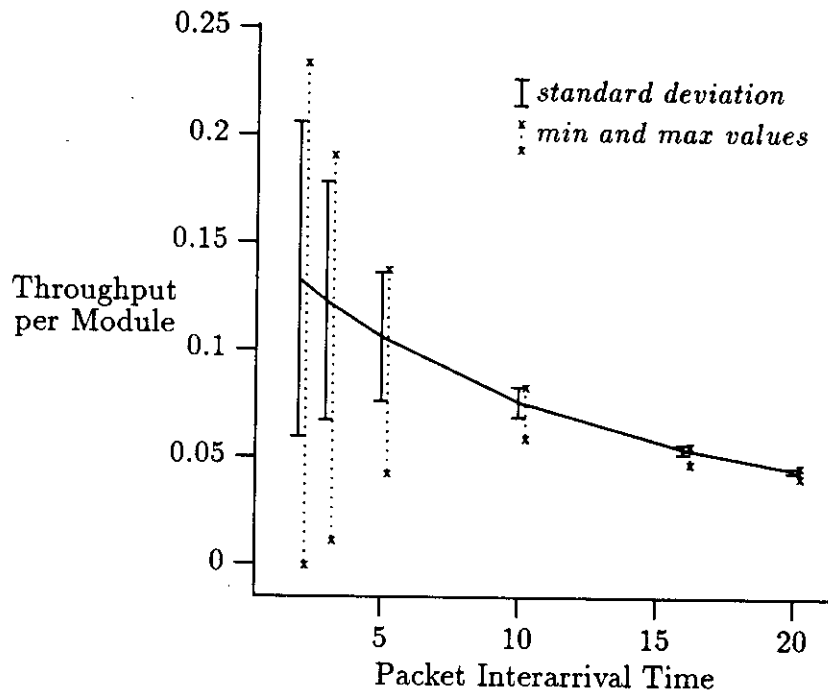


Figure 5.8: Throughput per Module versus Interarrival Time for GridNet

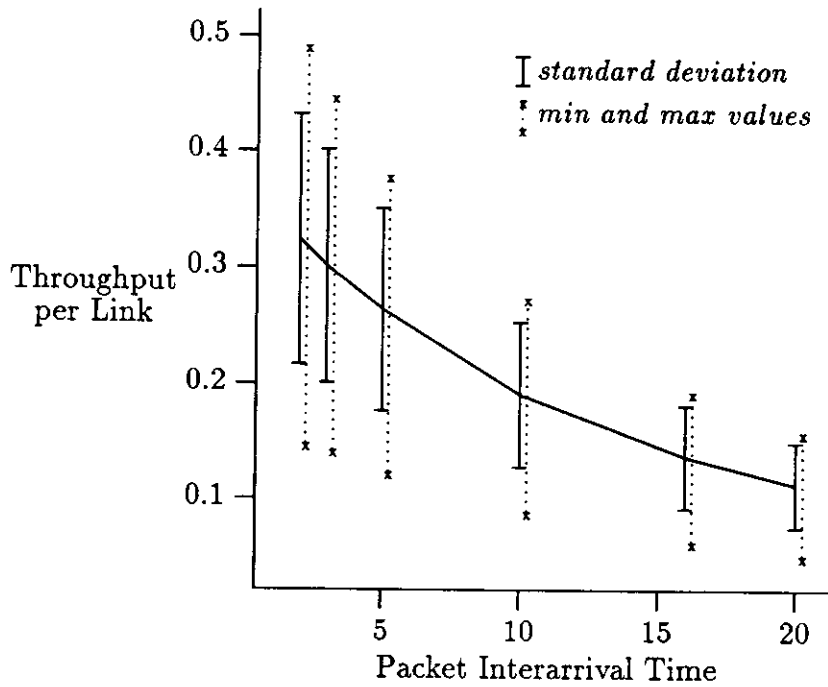


Figure 5.9: Throughput per Link versus Interarrival Time for GridNet

encouraging. For time this limit is defined as

$$t_{limit} = t_{sync} + t_{hop} \cdot \bar{h}$$

where,

- t_{sync} is the synchronization time from the arrival of the local packet till the next frame tick. It is derivated in Chapter 6,
- t_{hop} is the time it takes for each hop, which is equal to the frame time ($t_{hop} = 1$ by assumption),
- \bar{h} is the average hop number,

$$\bar{h} = \frac{1}{N \cdot (N - 1)} \sum_{\text{for all } i, j, i \neq j} d_{ij},$$

where N is the number of nodes and d_{ij} =distance from i to j . We are assuming that all nodes communicate with each other with a uniform distribution.

The throughput limit is defined as:

$$Tp_{limit} = \min(\bar{\lambda} \cdot N, \frac{L}{\bar{h}})$$

where,

- $\bar{\lambda}$ is the average packet arrival rate to a node,
- L is the number of links in the network,
- \bar{h} is the same as above.

Since the link capacity is one packet per frame, the upper bound for the throughput is L/\bar{h} .

The time behavior can be explained by the high latency of this protocol in both the initial access (local) and change in direction (switched). For example, for $\lambda = 1/5$, $t = 12$ while $t_{limit} = 4.3$.

Regarding throughput, we noticed that at high load some nodes have hardly been able to send any packets. Hence there is a fairness problem with this scheme. Also some links were used almost 50% of their capacity while others were hardly utilized at all. An important lesson was learned: it is not only the topology that influences the delay, the protocol has a major role.

Another drawback of this scheme is that if the load is not uniform, there could be congestion and possible buffer overflow. Hence, packets could be lost. This is because the access protocol is based on fixed assignment, and does not adjust to communication demands. Even with balanced uniform load, link utilization varies over a wide range (see Figure 5.9). This means that some of the links are underutilized, and synchronization time is too high when there is a need for change in direction.

In the next section, we will introduce a new protocol to achieve better results. The goal was to develop protocols with higher performance, which better adapt to different load conditions, and have a fairer behavior.

5.2 GridNet/p2 - A Different Protocol

The main reasons for the modifications to the GridNet protocol were:

- In the earlier scheme, links were not fully utilized and had great variation in terms of throughput,
- While the protocol was providing equal bandwidth on a link basis, it was compromising the overall network fairness. This occurs because
 - service is shared by locally generated and switched traffic,
 - switched traffic has priority over local, and
 - the pattern of the switched traffic is not homogeneous across the topology. Therefore, some switches have their local traffic unfairly penalized.

The objective for this change, was to find an alternative that was more “fair”, (equitable) and provided similar throughput for all modules at uniform load condition. We noticed that at high load some switches in GridNet were partially shut out (see Figure 5.8). The reason is that at polling time, they had to dedicate the frame to a packet changing direction (switched packet) rather than to a newly generated (local).

For instance, if we look at Figure 5.1, we notice that all traffic in *switch*₁ that comes from or through *switch*₅ and is not destined to *switch*₁, changes direction in *switch*₁. Note that this is the case for most traffic. Therefore, in heavy load, a continuous stream of packets may come from *switch*₅ into *switch*₁. Since a frame is assigned to *switch*₁ to transmit (switched or local packet) every three frames (as determined by the polling cycle), above a certain load condition, there will be

no frame available for local packets at *switch*₁. Other corner switches are also affected: *switch*₄, *switch*₁₃, and *switch*₁₆.

The following paradigm will explain why one should give higher priority to an *on going* packet rather than to a new. A new packet will become an *on going* packet from the next switch on and thus, it will be serviced faster from that point on. If, on the other hand, the new packet had higher priority it would get service sooner. However, from the next switch on, it will have to wait/defer to a new one, since it would now be an *on going* packet. This mechanism on a high load situation (or a load beyond some threshold) will lead the network to a congested state. Hence, *on going* packets should have higher priority. The solution lies in controlling the flow before it enters the subnetwork in order to achieve a more fair access with better results of delay and throughput. The same strategy (i.e. giving precedence to transit traffic over input traffic) is commonly employed in flow control of conventional computer networks. An example is found in the input buffer limit policy [GK80].

5.2.1 Protocol

The basic principle of this proposal consists in shifting the polling function from the frame generator to the individual switches. In the previous scheme, there was no attempt to estimate how much bandwidth was available to a switch for local or switched traffic. The amount of bandwidth available for new packets in a switch was primarily determined by its topological position.

The idea was to implement a token-less polling scheme in order to avoid the overhead and latency of a token scheme. The result was a distributed input rate control (DIRC) protocol, where switches would be assigned frames in an orderly and periodic fashion.

The fundamental changes are: the frame generators create frames marked as empty, as they start their journey down the line; and the switches control the access to the frames by keeping count of how many frames have passed since their last transmission on that line. The basic idea is to have the switches wait for a number of frames before attempting to transmit.

A key role in the GridNet/p2 protocol is the counter, which controls how often a node can place a new packet in the sub-network. Initially the counter is set to *FC* (frame count) and is decremented each time a frame passes by. When the counter drops to zero, the node may attempt to transmit the local packet in the first empty frame. However, transmission of switched packets takes precedence. Also, frames are marked as free when packets reach their destination. The protocol can be summarized as follows:

```

initially counter  $\leftarrow FC$ 
at frame tick do
  if counter  $\neq 0$  then
    counter  $\leftarrow$  counter - 1
  fi
  if frame is free then
    if switched_buffer not empty then
      transmit from switched_buffer
    else if counter = 0 and local_buffer not empty then
      transmit from local_buffer
      counter  $\leftarrow FC$ 
    fi
  fi
od

```

We claim that the protocol enjoys the liveness property given that $FC \geq FC_{min}$ to be later defined. To fulfill this property, it must be deadlock and lockout free. No deadlock can occur because there is no contention for resources that creates cyclic waits, i.e., a packet changing direction (switched packet) or local packet just waits for an empty frame (which is independently generated) following the rules dictated by the protocol.

Lockout free in this context means that no packet would wait indefinitely for a free frame. To satisfied this, we have to find the value of FC_{min} which depends on the routing table and traffic pattern (load condition). The value of FC must be chosen according to how many packets might contend for a link and this can be tuned to the average or to the worst case scenario. The difference here is the number of buffers necessary for the switched packets.

Analyzing a 16 node GridNet, we noticed that the worst case is 15 connections sharing a link. Therefore, if a switch is allowed to place a new packet at least every 15 frames, no lockout will occur. For instance, this would happen if all switches send packets to *switch*₂ (see Figure 5.1). In such a case all packets would have to pass from *switch*₁ to *switch*₂ via a single link. Therefore with $FC = 16$ (FC is first decremented and than compared to zero), we are basically serializing the shared access for this link.

This is indeed a very special case (really the worst possible case). For such a case, an altogether different approach should be taken (i.e. a different protocol for instance). In a uniform load, FC can be less than 16 because, on average, no link will be contended by 15 origin-destination communication pairs. By chosing a lower FC , throughput will be higher because switches can insert local packets with a lower waiting time.

From the network point of view, a lockout condition is less serious than a deadlock; the former deprives part of the network, while the latter brings the network to a halt. However, since on top of the network there are still other layers of functionality, one must be careful not to allow a key function on a node (e.g. a scheduler) to be locked-out.

5.2.2 Results

We start by comparing GridNet and GridNet/p2 in terms of delay and total throughput. This is seen in Figure 5.10 and 5.11. The frame count (FC) for this comparison is equal to 8. We note that the delay is greatly improved, but

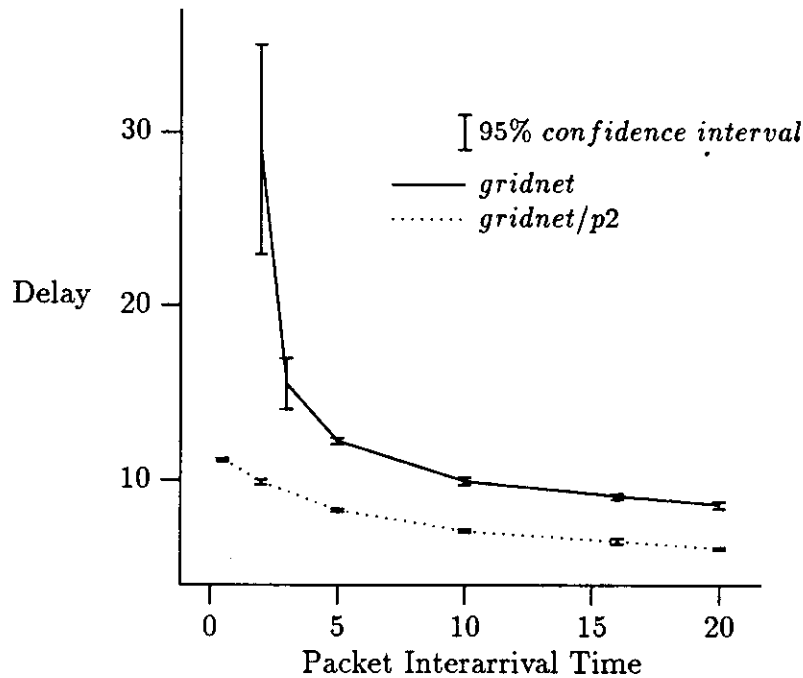


Figure 5.10: Delay versus Interarrival Time

the total network throughput remains about the same.

In the next three graphs (Figures 5.12, 5.13 and 5.14) we show how the FC influences the performance. With $FC = 1$, we have the case when nodes may send a local packet whenever a free frame comes along, and the switched buffer is empty. With $FC = 16$, the switch for that particular line, will wait for 15 frames to come by, before any attempt to transmit is made. This explains why the delay is so high and the throughput so low. In Figure 5.14 we compare the power versus the packet interarrival time. Power [Gai83] is a function that captures the tradeoff between two conflicting performance measure: delay (T) and throughput (γ). It is

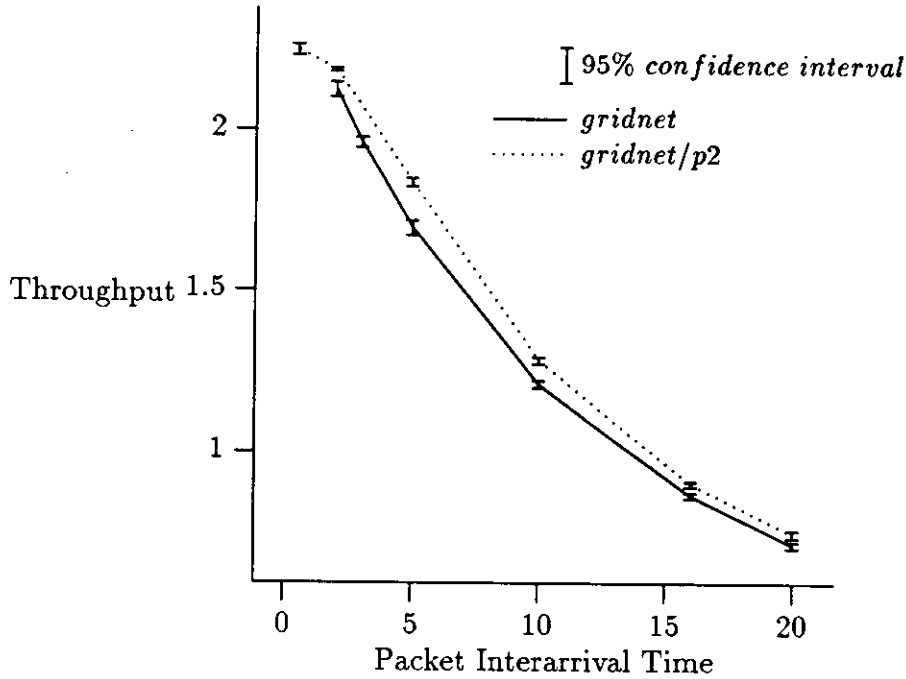


Figure 5.11: Throughput versus Interarrival Time

defined as their ratio: $P = \gamma/T$. We may notice that the maximum power for the different FC values occurs within the same range of packet interarrival time.

In Figure 5.15, we may observe the effect of FC in terms of fairness. The graph shows coefficient of variation in throughput across the switches versus the total throughput of the network. The coefficient of variation C_X is defined as σ_X/\bar{X} , where σ_X is the standard deviation of the throughput of the switches and \bar{X} its average. Basically this measure shows how much variation there is for the different values of FC . A low coefficient indicates a fair access. We note how “unfair” the previous protocol (GridNet) was, and; how GridNet/p2 one behaves as FC decreases.

The above results were obtained for uniform load condition. For a non uniform load condition, depending on the value of FC , packets could accumulate in certain switched_buffers and cause overflow.

Table 5.1 shows the average link utilization efficiency, which is the ratio between the actual average link utilization and the theoretical maximum, in high load condition. We note that GridNet/p2 yield much better utilization than GridNet for low FC values, as expected.

The sensitivity of performance to the value of FC and the interdependence of FC and traffic pattern suggest the value of FC might be individually adjusted for each switch according to load conditions. However, these conditions have to

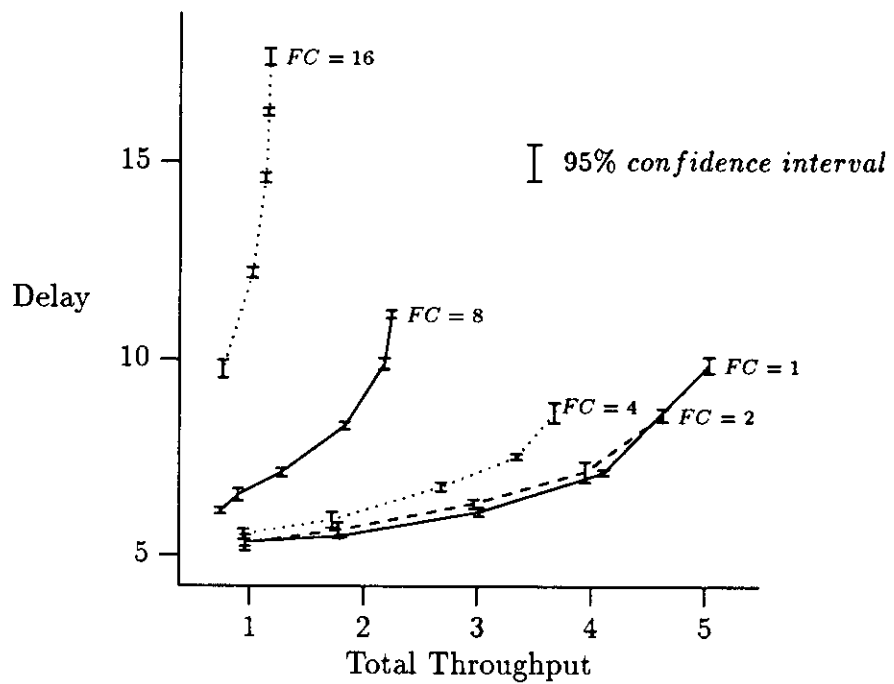


Figure 5.12: Delay versus Throughput

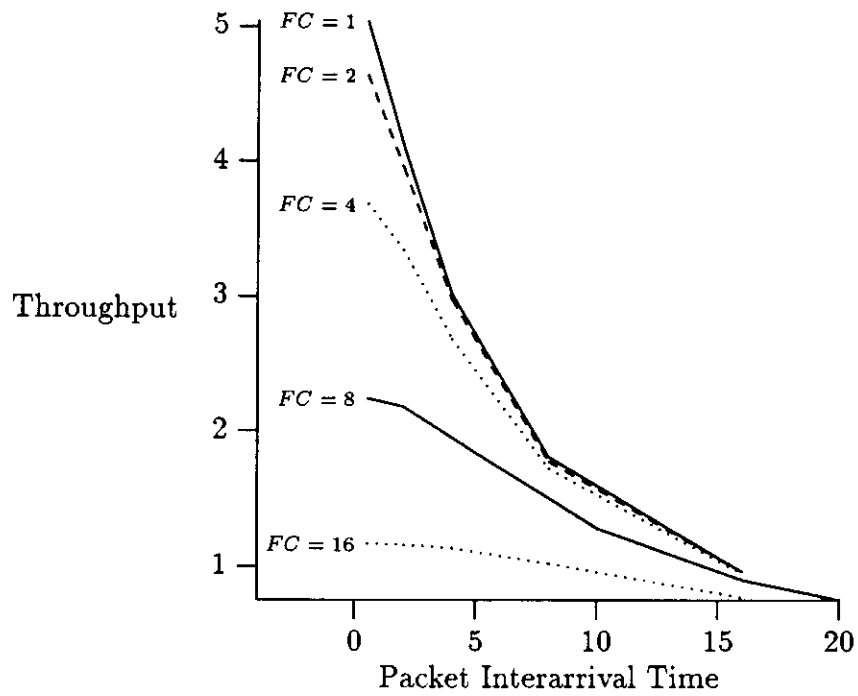


Figure 5.13: Total Throughput versus Packet Interarrival Time

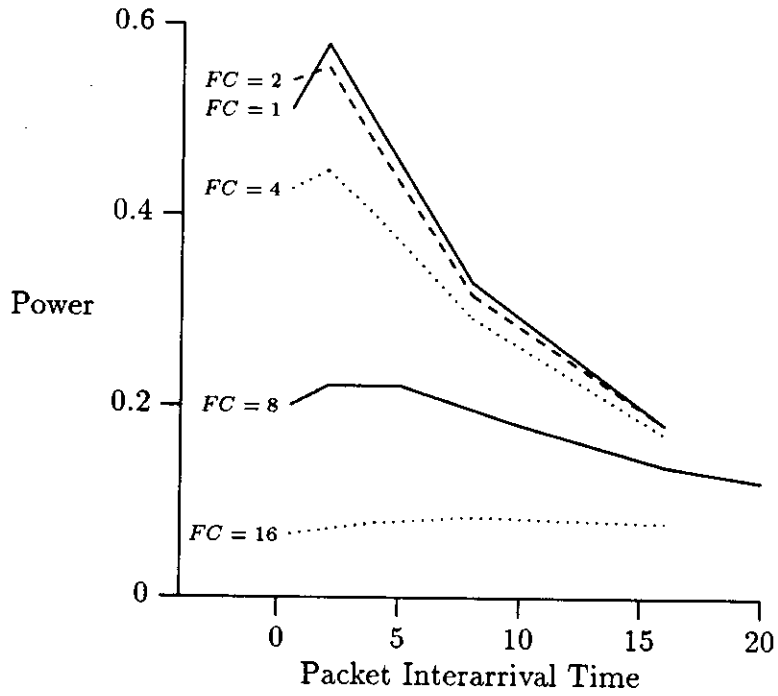


Figure 5.14: Power versus Packet Interarrival Time

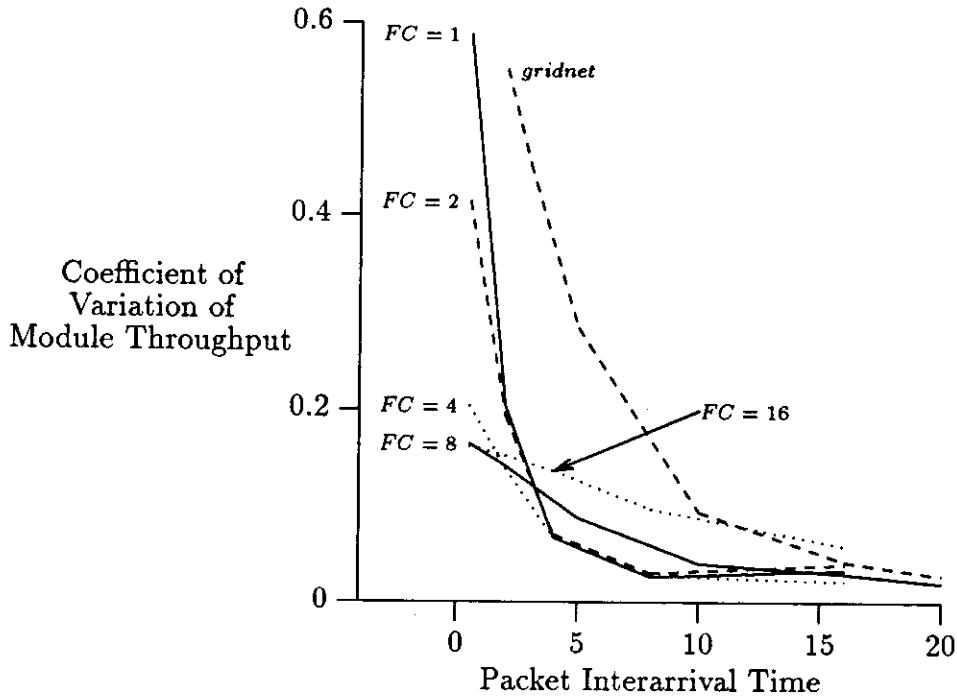


Figure 5.15: Coefficient of Variation versus Packet Interarrival Time

GridNet		0.34
GridNet/p2	$FC = 1$	0.80
	$FC = 2$	0.73
	$FC = 4$	0.58
	$FC = 8$	0.35
	$FC = 16$	0.19

Table 5.1: Aggregated Link Utilization Efficiency

be known in advance and the correspondent FC s computed accordingly, since GridNet/p2 is not an adaptive protocol with respect to flow control. Namely, the protocol performs input flow control based on predefined values of FC .

In the next section, we will introduce changes to both topology and protocol in order to improve performance. The basic idea in terms of the topology was to come up with solutions that overcame the problems at the boundary of the mesh. As for the protocol, the goal was to develop a protocol that adapts better to different load conditions, and has fairer behavior.

5.3 RingNet

5.3.1 $2in - 2out$ Topologies

One of the shortcomings of the GridNet topology is that switches have different degrees of connectivity, depending on their topological positions. The connectivity varies from one input and one output to two inputs and two outputs. From Figure 5.1 we observe that $switch_{1,4,13,16}$ are $1in - 1out$, $switch_{2,8,9,15}$ are $1in - 2out$, $switch_{3,5,12,14}$ are $2in - 1out$, and $switch_{6,7,10,11}$ are $2in - 2out$. These different degrees of connectivity create bottlenecks that impact negatively on the total network capacity (performance). They also affect fairness, since the switches with $2out$ could have more bandwidth available than the ones with $1out$.

The topologies that we will describe below are based on the $2in - 2out$ concept for all switches and follow the guidelines for the on chip network described in previous chapters.

5.3.1.1 MRing Topology

The MRing topology is seen in Figure 5.16. Its name comes from the multiple rings that the interconnecting mesh creates. The end of the line switches (from GridNet)

are connected to one of the neighbors, thus, providing the $2in - 2out$ property. The frame-generators were replaced by links between neighboring switches. End of the line at $switch_5$ is connected to the origin of the line at $switch_1$ and so on.

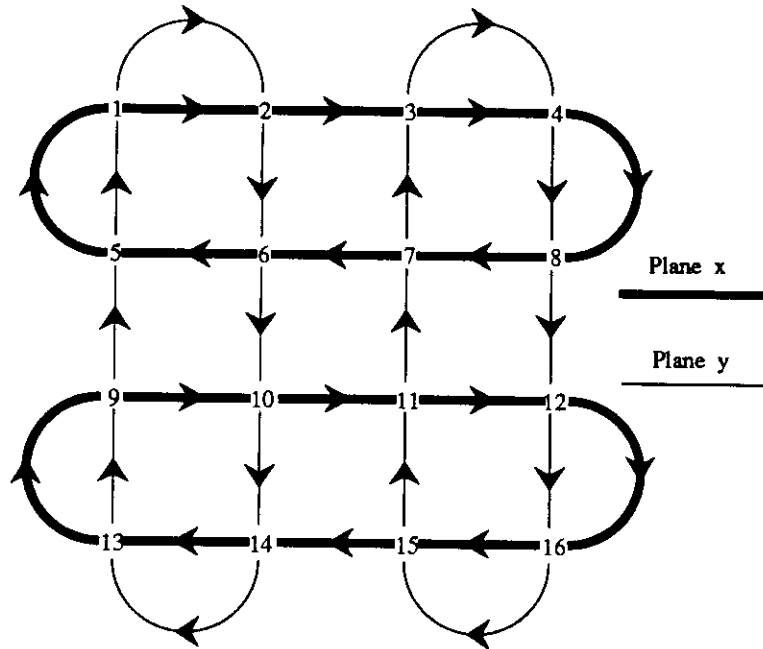


Figure 5.16: MRing Topology

The unidirectionality of the lines, now part of rings, is maintained. The timing function provided by the frame generator in the GridNet, can now be generated in a distributed fashion by self-timed circuits in the switches, and operate asynchronously, for example. Self-timed circuit design is a proven design methodology [Sut89, MBL⁺89, MC80] and will be specially applicable for systems where there is no central clock, like this one.

Another interesting characteristic of this topology is that some neighboring switches are connected by two links (the second link is the one that replaced the frame generators in GridNet). In order to distinguish them we separate them in two planes. Plane x (the orientation is decided by the elongation) consists by the rings formed by switches in the sets $\{1, 2, 3, 4, 8, 7, 6, 5\}$ and $\{9, 10, 11, 12, 16, 15, 14, 13\}$, while Plane y has $\{1, 2, 6, 10, 14, 13, 9, 5\}$ and $\{3, 4, 8, 12, 16, 15, 11, 7\}$.

The added links permit more possible paths between source destination pairs. This helps to distribute the load between the links. However this also increases the complexity of generating the routing table, which is dealt with in the next chapter.

5.3.1.2 DRing Topology

An alternative to the MRing which still satisfies the uniform connectivity condition is the DRing (double ring), presented in Figure 5.17. This option does not provide the same richness of possible minimum paths, and the average path length for a uniform load is higher than that for the MRing, however routing is simpler.

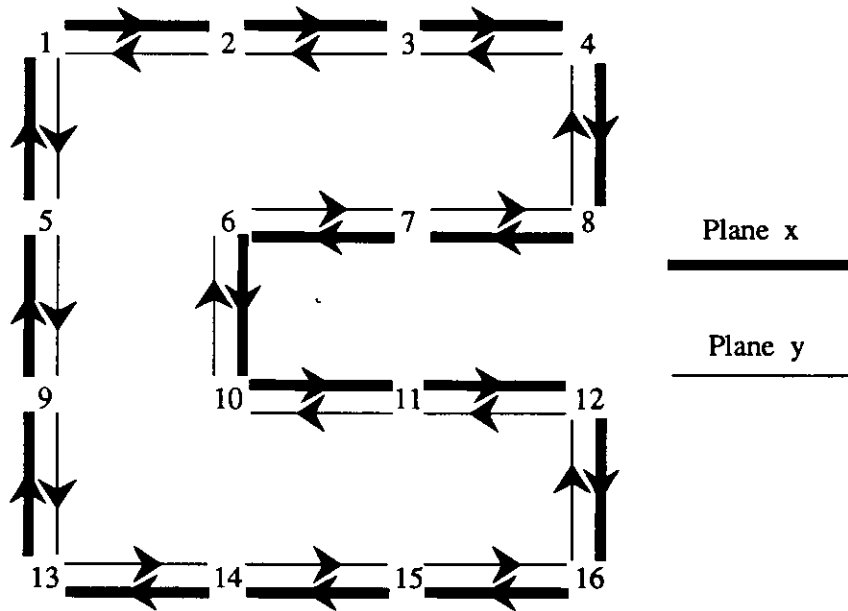


Figure 5.17: DRing Topology

5.3.1.3 Cubic Topology

The third (uniformly connected) alternative is the planar mapping of the hypercube of dimension 3 (16 nodes). This topology has been vastly covered in the literature [Sei85], and is present here to show that its planar implementation comes at a cost. In this topology, (Figure 5.18) links vary in length: some have length d and others have length $2d$, d being the link length in the previous topologies. Since distance is a major contribution in delay, this will ultimately compromise the performance. Furthermore, the wiring of the links has many crossings which are very undesirable in VLSI design.

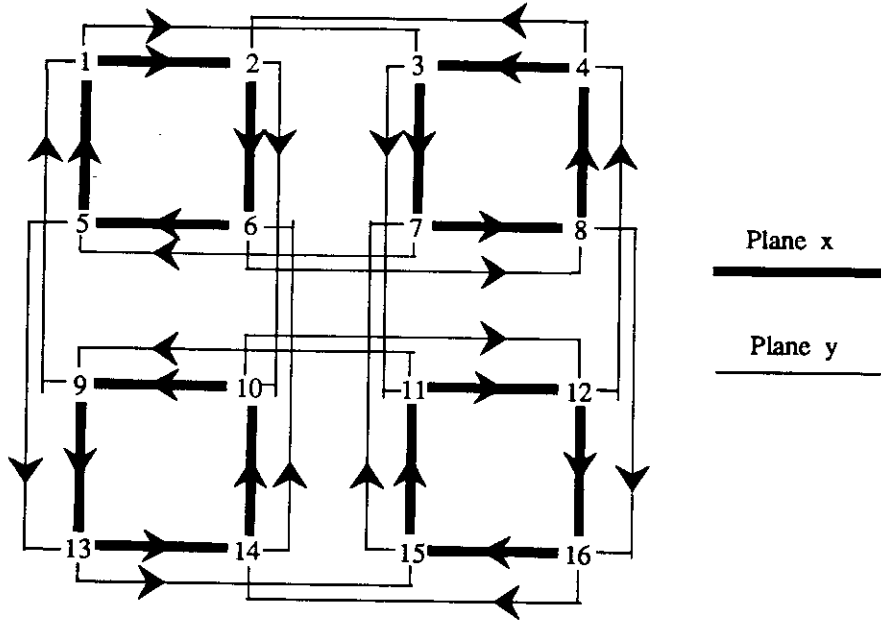


Figure 5.18: Cubic Topology

5.3.2 Protocols for Slotted Rings

In this section we discuss different access protocols for the kind of networks that conform to the VLSI constraint detailed earlier. The basic characteristics of these protocols environment are: 1) parallel interconnection among nodes (word wide expanded to include control signals), 2) slotted; the optimal long distance interconnection policy suggests the use of segmented unidirectional links (packets one bit long, word wide for low delay), and 3) cyclic structure (based on the $2in - 2out$ topologies).

We examine first the case of a single slotted ring with N nodes. This topology was first described by Pierce [Pie82] for a serial communications. However, since in our case we use a parallel link, that carries one word per slot we had to search for different solutions. The alternative approaches discussed here are of two types: token and token_less based. The token scheme is intrinsically pessimistic and usually “protects” the system from the worst case situation. On the other hand, token_less schemes are in principle more optimistic leading to lower delays at lower loads.

5.3.2.1 Token Based Access Protocols

In the explicit token scheme, one of the control signals has the function of token indicator, and is marked only in one of the N frames.

A node with a packet to transmit, waits for the token to come by and then uses the first free frame thereafter. Frames are marked empty by the destination node. The frame that has the token signal is used for data as any other frame. In heavy load, a node would have access on average every N frames (round robin). On light load, however, the average waiting for the token is $N/2$ frames with many empty frames not being used. The result is high latency. This is why we consider the token scheme as a very conservative scheme. Namely, this scheme is fair at high as well as light loads; however, at light loads, it cannot take advantage of unused resources to increase throughput and decrease delay.

5.3.2.2 Token-Less Based Access Protocols

A major difference between token and token-less schemes is that in the former the control information lies in the network (via the token) while in the latter, the nodes use some internal state to control the access to the network. We have devised two token-less schemes: DIRC (distributed input rate control) and DIRC_BP (DIRC with back pressure mechanism)

In the DIRC scheme the input rate is independently controlled by each node. Namely, the access is determined by the amount of time the packet has spent waiting. A counter keeps track of the waiting at each node. Initially or after each transmission, the counter is set to FC and is decremented with the passage of frames. When the counter reaches zero, the node may take the first free frame that passes by. Frames are marked empty at destination. If FC is set to $N + 1$ and the load is high this scheme accomplishes the same behavior as the explicit token scheme. In light load, however, the wait for the turn to transmit may be lower because nodes decrement the counter at each passing frame, independently of having a packet ready to send or not. Thus, a newly arriving packet may be able to go immediately. This scheme is the one used in GridNet/p2.

DIRC with Back-Pressure Protocol

In order to reduce the access time even further we developed DIRC_BP, where the basic idea is to let a node use the first free frame that comes by. This mechanism also increases the throughput significantly. Of course, precautions must be taken in order to provide a minimum service for all nodes, avoiding starvation, or deadlocks. To this end, if a packet has waited more than C frames, it can request a free frame

from the previous node (see Figure 5.19). The goal is to flow control the local traffic via a back_pressure mechanism. The request for a free frame is satisfied by a node as the highest priority of service, since this is how back pressure propagates. Hence, in terms of service priority, from the highest to the lowest, we have a) free frame request, b) through traffic, and c) local traffic. In the sequel, we present three different versions of a protocol (Protocol I, II, and III) based on backpressure.

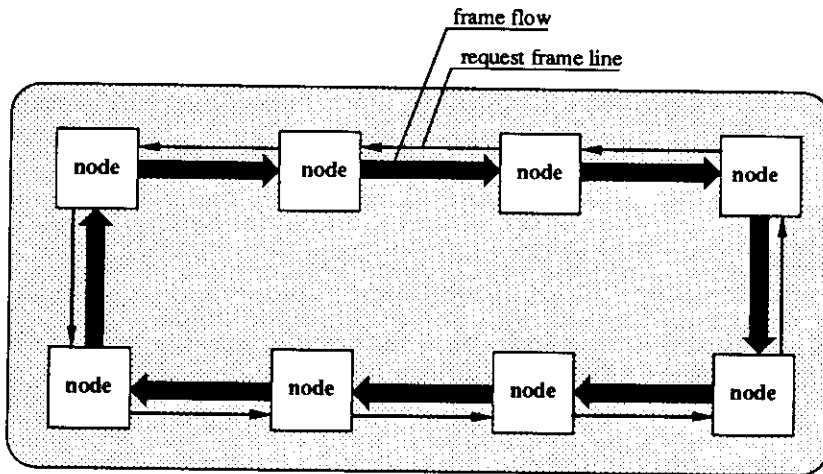


Figure 5.19: Cyclic Topology and Flow

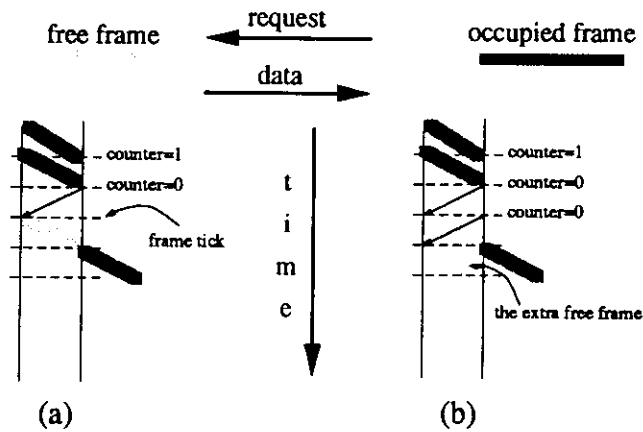


Figure 5.20: Timing and Flow of Data and Control

Protocol I

In this protocol, for control purposes there is a *counter* and a logical flag *alt*. The *counter* has the function to keep track of the waiting of a local packet for a free frame (local packets may only enter the network on free frames, no preemption). The flag *alt* is used to alternate the requests to the upstream node ($node_{i-1}$) when back pressure is to be applied. The reason for this alternating request is that it takes one time frame for the request to reach the upstream node ($node_{i-1}$) and another time frame for the free frame generated by the previous node to reach the node in question ($node_i$). In case requests were done in consecutive frames, two free frames would be generated by the upstream node, while only one was necessary. This affects the performance in high load, since an extra frame moves downstream with no packet. This is shown in Figure 5.20 b.

Back pressure, e.g. RF (request for frame) is applied when a passing frame is occupied and either (or both): *counter* reaches zero (because of a local packet waiting) or; a through packet was queued at this node ($node_i$) due to a request from the node downstream ($node_{i+1}$). And as explained earlier, the request is sent on alternating frame times (seen in Figure 5.20 a).

Algorithmically, the access protocol can be described as:

```
initially, counter ← C, alt ← TRUE
at frame tick do
  if packet arrived then
    buffer it local_in buffer
    mark frame as free
  fi
  if counter ≠ 0 and local packet waiting then
    counter ← counter-1
  fi
  if frame requested then
    buffer incoming packet (if any)
    release free frame
  else if exists a through packet then
    (* either buffered or from the incoming frame *)
    send it
  else if local packet waiting then
    send it
    counter ← C
  fi
  if alt = FALSE then
    alt ← TRUE
  else if counter = 0 or through_buffer > 0 then
```

```

    request frame from upstream node
    alt ← FALSE
  fi
od

```

Properties of Protocol I

Theorem 1 *Protocol I is deadlock free.*

Informal proof: The proof consists of showing that no cycle for free frame requests is created. Namely, because of the alternating of the free frame requests, a node may request at most half of the frames from the upstream node (see Figure 5.20 a). The rest of the frames can be used to carry packets forward. In other words, there is always traffic flowing towards its destination. When the destination is reached, the frame will become free, ready to be used again.

A request propagates along the opposite direction of the flow till it finds a free frame (Figure 5.21). This will happen whenever the request reaches a packet that just got to its destination, or the previous node has sent forward a free frame because it had no packets for it (either local or through). Since there is always some packet which has arrived at its destination, eventually the request will be satisfied. □

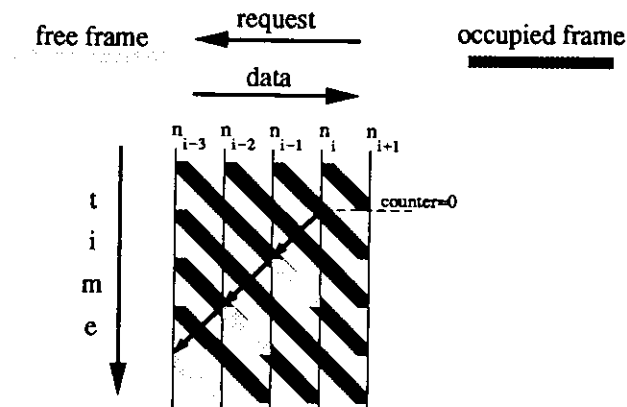


Figure 5.21: Flow of Free Frame Requests

Theorem 2 *Only one buffer is needed for through traffic.*

Informal proof: A node, as soon as it receives a request, forwards a free frame, and queues the through packet just arriving at the node (in case there is a packet and the packet is not destined to this node). At the same time, this node requests a free frame from the previous node. This is illustrated in Figure 5.22. By the time an eventual next request would be coming to this node, a free frame will be arriving (per its previous request). In conclusion, no additional packets will be queued at a node. \square

The result of this theorem is highly desirable for VLSI, where resources are “precious.” It also makes the buffer control mechanism much simpler.

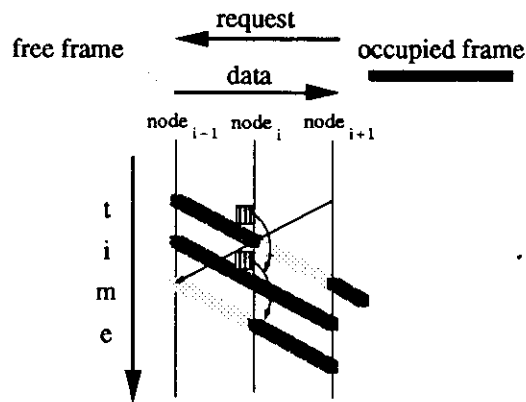


Figure 5.22: One Buffer Only

Function of the Counter

In a high load situation where all nodes are addressed, and with the counter with a high value ($C \gg N$), a node will only get a chance to transmit when a packet destined to it arrives, i.e., the packet is “consumed”, freeing up the frame which is immediately taken by the local waiting packet. Usually, no node will have to defer “this” free frame since there is no frame request because no counter reaches zero.

The function of the counter is to distribute (“spread”) the free frames generated by arrivals at other nodes, so that nodes that are not addressed so often get a chance to transmit. This is the basic function of the upstream frame request, that will force a “popular” destination node to defer transmission of its own and pass the free frame downstream to a waiting node.

C has critical impact on the performance of the network. If $C = 0$, (again on high load), due to alternating of requests, only half of the frames would be utilized. The other half is accounted as free frames forced by requests, since all nodes have

a local packet ready and the counter is always zero. Packets advance in a stop and go motion. This of course results in a high overhead with 50% of the frames not being used. If C is too high, it could mean that a node with low arrival rate might wait for a long period of time. When $C = N$, the network is “biased” towards one local access at least every N frames.

However, there are some problems with this protocol in certain conditions (unusual conditions, though) where a node might “starve” due to a bad choice of C . Consider the case where $node_N, node_{N-1}, \dots, node_2$ send packets to $node_1$ and all are always ready to send. Assume $C \geq 2$ for all nodes. In steady state, $node_N$ will be able to transmit at least half of the frames because downstream requests for free frame are alternated between and all frames arriving from upstream are free. $node_2$ will get $1/C$ (every C frames, it requests a free frame from $node_3$ and there are no requests from $node_1$ to $node_2$). $node_3$ will get $1/C$ if $2/C < 1/2$ otherwise it will get $1/2 - 1/C$ and $node_{4toN-1}$ will starve. (i.e. they will get zero throughput) $node_4$ will get $1/C$ if $3/C < 1/2$ otherwise it will get $1/2 - 2/C$ and $node_{5toN-1}$ will starve. and so on till $node_{N-1}$ will get $1/C$ if $N - 1/C < 1/2$ otherwise it will get $1/2 - (N - 2)/C$.

Regarding the through packet queue, the request for a free frame can be changed so that it is triggered when queue is B rather than 1 as described before. This will allow the network to absorb a greater variance of traffic without nodes requesting free frames so often. This way, the number of free frames moving on the network due to requests is reduced, and the throughput performance improved. If requests are issued when the through buffer has B packets, it also follows that the buffer size of B at each node is sufficient so that no packets are lost.

The possible starvation condition on Protocol I lead us to revise this protocol and propose two variations (Protocol II and III).

Protocol II

The difference between Protocol I and Protocol II is that now we define an “accumulation” state when *counter* is zero and at the same time there is a request for a free frame and there is a packet passing through. When this state is reached, the restriction to alternate the requests from the upstream node is bypassed and request are made in consecutive frames.

The algorithmic description for Protocol II follows:

```

initially, counter  $\leftarrow$  C, alt  $\leftarrow$  TRUE

at frame tick do
  if packet arrived then
    buffer it local in buffer
    mark frame as free
  fi
  if counter  $\neq$  0 and local packet waiting
    and no frame request from downstream then
    counter  $\leftarrow$  counter-1
  fi
  if frame requested then
    buffer incoming packet (if any)
    release free frame
  else if a through packet is present then
    (* either buffered or from the incoming frame *)
    send it
  else if local packet waiting then
    send it
    counter  $\leftarrow$  C
  fi
  if (counter = 0 and through_buffer > 0)
    or through_buffer > 1 then
    request frame from upstream node
    alt  $\leftarrow$  FALSE
  else if alt = FALSE then
    alt  $\leftarrow$  TRUE
  else if counter = 0 or through_buffer > 0 then
    request frame from upstream node
    alt  $\leftarrow$  FALSE
  fi
od

```

For this protocol, we have traded the starvation (lockout) of nodes for a possible deadlock condition, therefore the value chosen for C is of utmost importance. The reason is that now there exists a condition where traffic will not progress.

Suppose one node reaches accumulation. This will imply that two consecutive free frame requests will be issued. It is a possible scenario that other requests will “align” on the path of requests of the original request, thus creating an accumulation of requests. It can be shown that if these requests are not satisfied by packets arriving at their destination or by unused frames, they will eventually

loop around and reach a node which is itself requesting a free frame. This creates a cycle of requests, therefore a deadlock where all nodes are sending free frames to the upstream node and at the same time, requesting a free frame from the one downstream. Meanwhile, the through packets, as well as the local packets will stay where they are. This deadlock situation was reproduced in the protocol simulation.

A request for free frame once put in motion, will only stop (be absorb) at a node in its upstream travel on the condition that 1) that node has neither a through packet, 2) nor a local packet ready ($counter \neq 0$). Condition 1) may also be caused by an arrival at that node. Otherwise the request will propagate. As it propagates, it will not allow some other nodes to transmit and as they reach $counter$ zero, a wave front of requests starts forming.

We now want to find out what is the value for C_{min} that the counter should be set after a local packet is transmitted, which will make the protocol deadlock free.

First, we consider the case where nodes send packets at the same time and all packets have the same OD (Origin to Destination) distance. Also, we assume that the nodes always have local packets ready to be sent out. Figure 5.23 illustrates a four node ring with packets with longest OD distance.

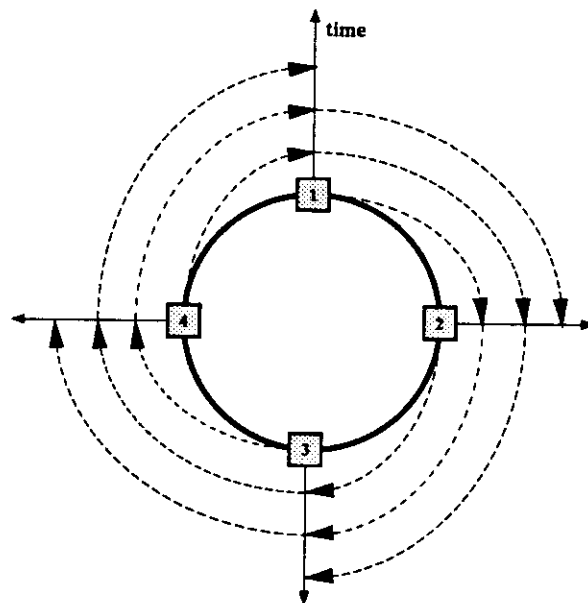


Figure 5.23: Progression of Packets in a Lock Step Case

One can show that Requests for Free frames (RF's) will not occur if the counter at the nodes is set with $C = C_{min}$, $C_{min} \geq N - 1$, where N is the number of nodes in the ring.

The C_{min} value is based on the fact that the longest OD path is $N - 1$. For instance, suppose that all packets have path with $OD = N - 1$. If $C < N - 1$, the counter at each node will expires before a packet destined to that node arrives (which would provide the free frame for the transmission of the local packet). Noting that each node has a through packet coming (which will not be arriving to its destination on this time frame) and that $counter = 0$, a RF will be issued. Since this happens to all nodes at the same time, all nodes in the next time frame will buffer the through packet, send a free frame downstream and RF upstream; and this will continue forever. Thus, no data traffic will progress.

If $C = N - 1$ ($C = OD$), the counter expires exactly at the same time that there is a packet arriving to its destination. Therefore, there is no RF since the arriving packet will generate the free frame needed.

For $C > N - 1$, the $counter$ does not expires. The reason is that there will be a packet reaching its destination (for any of the nodes) in at most $N - 1$ and when this happens, this frame will be used by the local packet, which in turn will set the $counter$ back to C .

We now want to consider the more general case in which we relax the synchronous transmissions and uniform OD distance. We introduce the notion of RFW (Request Frame Window) and DW (Data Window). RFW is the wave front of request and DW is the “space” left where frames may carry packets forward. When the system is “near” a deadlock condition, a cycle is formed by one RFW and one DW . Let RFW_w be the width of the RFW and DW_w the width of DW . Clearly, $RFW_w + DW_w = N$, where N is the number of nodes.

In order not to have deadlock, we have to choose C such that the RFW (Request Frame Window), does not become too wide, reducing the DW (Data Window) to zero. See Figure 5.24.

In order to proof that this does not occur (i.e. no deadlock), we use the following conjecture:

Conjecture 1 *There are at least two packets that arrive to their destination (i.e. absorbed) when $DW_w \geq 3$.*

Since the throughput of the ring is at least one per time frame, this conjecture is clearly true. From the Figure 5.24 we note that the first frame “band” is used for sending free frames, while the rest may carry frame with packets.

We should also note that the $counter$ is not decremented if a node receives a request frame. Hence, $counter$ values do not change during RFW and no requests can start during RFW .

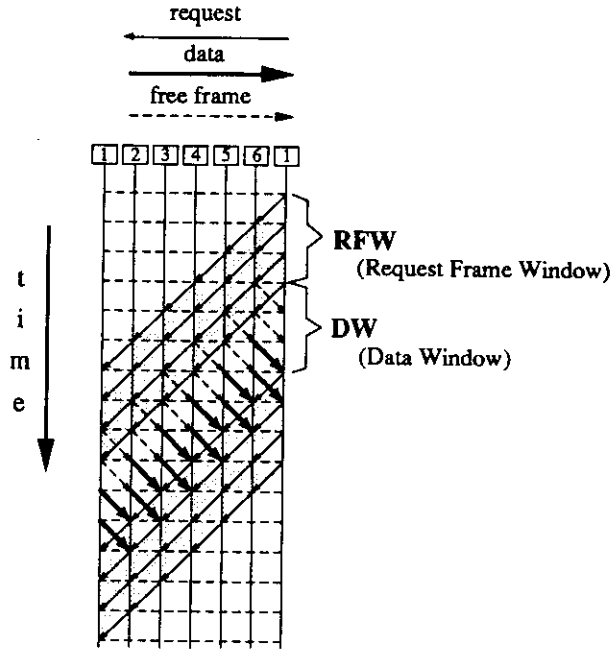


Figure 5.24: *RFW* and *DW* Windows

An important observation is that when $DW_w = 3$, any packet arriving to its destination will cancel an RF. In fact, a free frame is generated and the request will no longer propagate upstream. Also, when $DW_w = 3$, any data frame advance will reach a node which is servicing a request. Hence, any packet reaching its destination in such DW will consume the RF. This can be seen in Figure 5.24.

If $RFW_w = N - 3$ (which also implies in $DW_w = 3$), this means that $N - 2$ nodes have requested frames and have received free frames. From Conjecture 1, we know that with $DW_w = 3$, at most 2 *counters* can expire in this window. Therefore, the minimum value for the setting of the *counters* is $C_{min} = (N - 2) + (2) = N$, where the first term represents the number of nodes whose *counters* have expired and that have been satisfied, and the second term stands for the maximum number of nodes whose *counters* might expire. Otherwise, DW will shrink to a size where no more frame with packets will progress. \square

Extensive simulation experiments were carried out to verify this property. In a ring with $N = 8$, it was found that no deadlocks occur for $C = 8$. For $C = 7$, on the other hand, the system would consistently deadlock.

It was earlier mentioned that Protocol II only needs a limited buffer for the through traffic. We now want to be more specific about this property and prove in Theorem 3 that only two buffers are required.

Theorem 3 *The buffer size for through traffic is two.*

Informal proof: Suppose $node_{i+1}$ reaches $counter = 0$ and receives a RF from downstream at t_k (e.g. $node_{i+1}$ is in “accumulation” state). At t_{k+1} , $node_i$ will receive a RF and will itself send an RF to $node_{i-1}$. At this point, $node_i$ will have one packet in the through buffer. The only other packet that may get to $node_i$ is the one sent by $node_{i-1}$ at t_{k+1} because its first RF will get to $node_{i-1}$ at t_{k+2} . $node_{i-1}$ from that time on will not send any other packet until $node_i$ stops requesting free frames. This is illustrated in Figure 5.25. \square

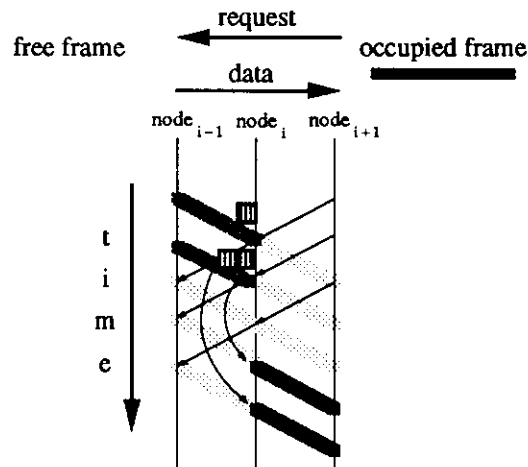


Figure 5.25: Two Buffers Only

Protocol III

This protocol differs from the previous one in that a node gets credit for the free frames it has given up to through traffic while it was its turn ($counter = 0$) and uses this credit to adjust the $counter$ for the next access.

Namely, the $counter$ now can become negative and when this happens, requests are issued in consecutive frames rather than on alternate frames as in Protocol II. The limited buffer size also holds for this protocol.

Bandwidth is now better distributed among the nodes because each node keeps track of frame counts that went beyond the threshold (C), and compensates by lowering the $count$ for the next packet.

However, this protocol is not deadlock free for small values of C . Since $counters$ are not frozen during requests for free frame, a number of nodes might start re-

questing frames during a *RFW*, which can eventually take over all frames, reducing *DW* to zero. In our experiments, when destination was chosen from a uniform distribution (average distance of $N/2$) we did not find a deadlock condition for $C \geq N + 2$. Yet, for longer destination distances, the condition could be as high as $C > 2N$.

On the other hand, deadlock detection is very simple. It is sufficient that nodes count the number of consecutives request for frame. If $counter(RF)$ reaches N , the appropriate actions should be taken.

```

initially, counter  $\leftarrow$  C, alt  $\leftarrow$  TRUE

at frame tick do
  if packet arrived then
    buffer it local_in buffer
    mark frame as free
  fi
  if local packet waiting then
    counter  $\leftarrow$  counter-1
  fi
  if frame requested then
    buffer incoming packet (if any)
    release free frame
  else if exists a through packet then
    (* either buffered or from the incoming frame *)
    send it
  else if local packet waiting then
    send it
    if counter < -1 then
      counter  $\leftarrow$  counter + C
    else
      counter  $\leftarrow$  C
    fi
  fi
  if (counter  $\leq$  0 and through_buffer > 0)
    or through_buffer > 1 then
    request frame from upstream node
    alt  $\leftarrow$  FALSE
  else if alt = FALSE then
    alt  $\leftarrow$  TRUE
  else if counter  $\leq$  0 or through_buffer > 0 then
    request frame from upstream node
    alt  $\leftarrow$  FALSE

```

5.3.2.3 Results for Protocols I, II, III

A comparative study of Protocols I, II, and III was done by simulation using RESQ. The models consisted of 8 nodes in a ring topology (i.e., $N = 8$, as in Figure 5.19). The results shown in Figures 5.26 and 5.27 are for $C = 10$ and assumed uniform destination of packets as well as poisson arrivals. We note that for these parameters, all protocols have the same level of performance. We did observe, however, that Protocol III deadlocked in a simulation experiment with $C = 9$ when all nodes were always ready to send packets (heavy load). Also, the delay for packets from generation to arrival is bounded due to input buffer limitation (finite population) and fair service.

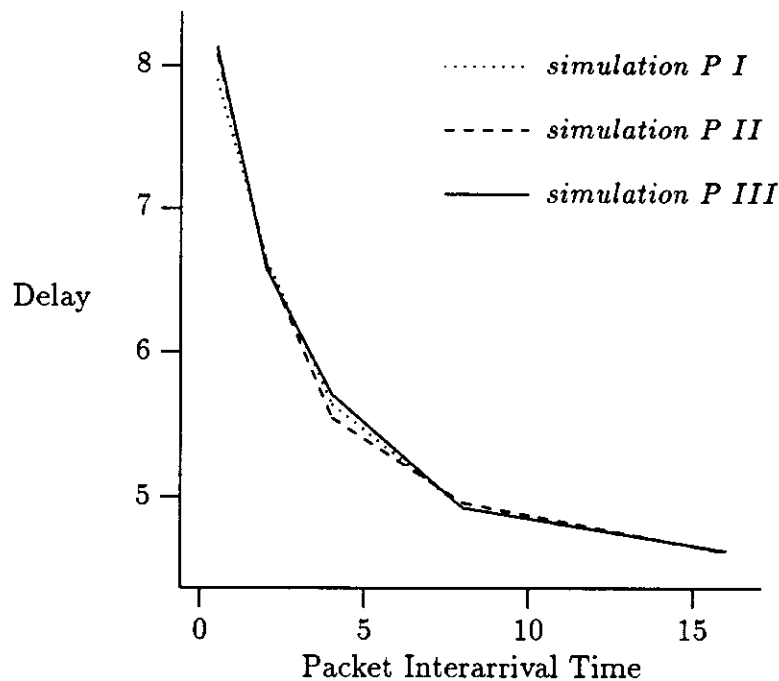


Figure 5.26: Delay versus Packet Interarrival Time

5.3.3 Protocols for RingNet

In any of the RingNet topologies, seen before as Mring, Dring and Cubic, a switch is positioned on all crossing points of the network and is responsible to provide

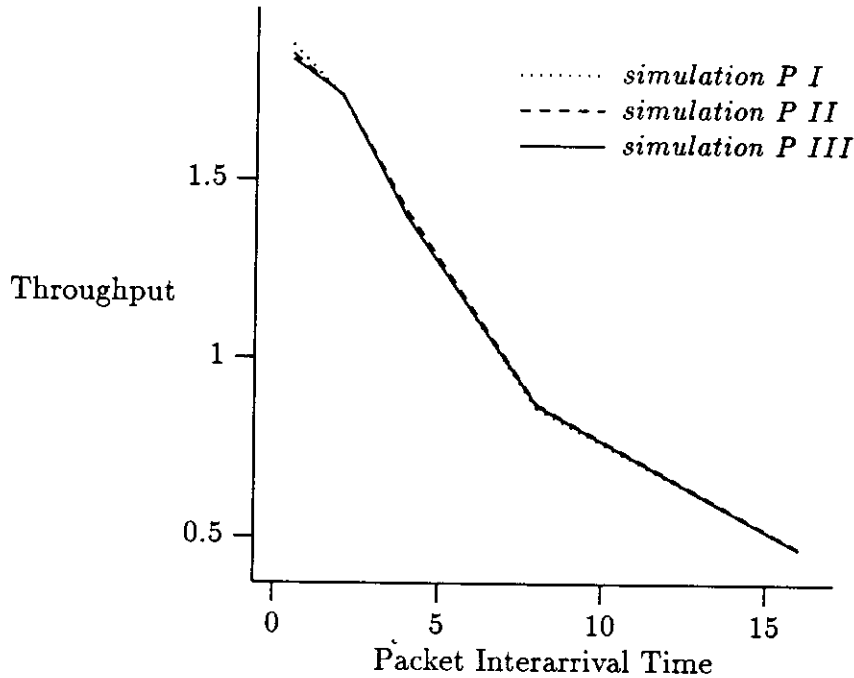


Figure 5.27: Throughput versus Packet Interarrival Time

the overall coordination of the traffic that goes on the network. The switch is composed of two nodes each one with a line handler (as seen in Section 5.1) that takes care of one of the planes, namely, x for the loop elongated on the x direction and y for the other loop). This is also shown in Figure 5.28. We refer to the traffic that changes direction (from x to y , for example) as switched traffic. The traffic that does not change direction is called through traffic.

The function of coordinating the access to all traffics types (local, through, and switched) is performed by the protocol which resided in the nodes.

5.3.3.1 Protocol RN-I

The node for the local and through traffic operates in a way similar to what described earlier for the ring topology. However, it acts as a gateway for the switched traffic. The protocol to perform this function is based on Protocol II and will be described later. We will consider the through traffic and the switched traffic as having the same priority, since what at a particular node is through traffic, could have been switched traffic earlier on.

In Protocol II, as well as here, only free frames in the plane where the node lies (coming from the previous node or arrivals) can be used for the packets waiting to go. Therefore, in order to get access, any of the waiting packets (namely,

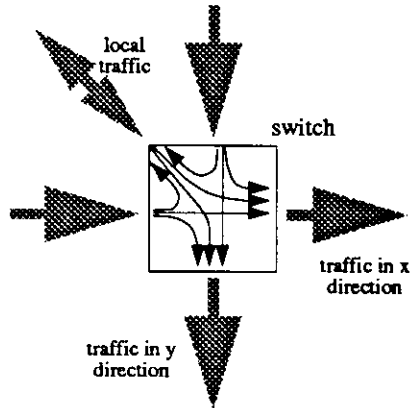


Figure 5.28: Flow of Data in a Switch

local, through, and switched) might have to rely on requesting free frame to the previous node, from where the through traffic comes from. Thus, in order to avoid congestion due to the switched traffic, an analogous scheme of backpressure is proposed.

Table 5.2 shows in what condition a request for free frame on the local plane is triggered. The decision is based on the local *counter*, the accumulation of through (*thr*) and switched (*sw*) traffic on the local plane and the accumulation of the traffic switched from this node to its orthogonal node (same switch, but the node on the orthogonal plane).

Condition	Action	
	rf_x	alt_x
$counter_x = 0$ and $sw_x + thr_x + sw_y > 1$	true	false
$sw_x + thr_x + sw_y > 1$	true	false
$counter_x = 0$ or $sw_x + thr_x > 0$ or $sw_y > 1$	true	true
$counter_x \neq 0$ and $sw_x + thr_x = 0$ and $sw_y = 1$	false	true

Table 5.2: Decision Logic on $Plane_x$

The protocol description follows:

initially, $counter \leftarrow C$, $alt \leftarrow true$

at frame tick **do**

if $counter \neq 0$ and local packet(s) waiting **then**

$counter \leftarrow counter - 1$

if frame requested **then**

```

    release free frame
else if there is a through or switched packet then
    send it
else if local packet ready then
    send it
    counter  $\leftarrow C$ 
fi if (counter = 0 and  $thr_x + sw_x + sw_y > 1$ )
    or ( $thr_x + sw_x + sw_y > 2$ ) then
    request frame from previous switch
    alt  $\leftarrow true$ 
else if alt not true then
    alt  $\leftarrow true$ 
else if counter = 0 or  $thr_x + sw_x > 0$ 
    or  $sw_y > 1$  then
    request frame from previous switch
    alt  $\leftarrow false$ 
fi od

```

5.3.3.2 Protocol RN-II

The second alternative is an alternating protocol which is based on the premise that half of the bandwidth should a priori be reserved for traffic coming from x and the other half for the traffic coming through y . In this protocol, as opposed to the previous one, there is no distinction between the two planes. Figure 5.29 illustrates the concept of a node for this protocol.

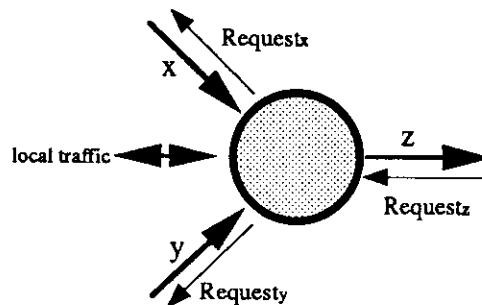


Figure 5.29: Node Traffic Flow

The protocol alternates between mode x and y . For instance, in mode x , it first checks for through traffic in x . If any, the node sends it. In case there is none, y is checked and sent if there is through traffic there. Finally, it checks for local packets. After this, the protocol goes on to check y , and so on.

Local traffic also alternates so that requests for free frames are issued to both incoming links. When *counter* reaches zero, the local packet is sent out on the link indicated by *alt_local*. It goes before any through traffic (since it has already waited the count period).

Back_pressure may be applied to both incoming links and is mode dependent. For example, in mode x if $\text{queue.length}(x) > 0$, a request for free frame is issued on link_x . However if $\text{queue.length}(y) > 1$, a request is issued on link_y . And similarly for mode y . The first test generates the alternate requests that guarantee that half of the bandwidth, while the second test permits consecutive request in case there is accumulation. This protocol, like Protocol II, needs a limited buffer for the through traffic, though in this case, the number is four because of the merging of two input streams.

The details of the protocol follows.

initially, $\text{counter} \leftarrow C$, $\text{alt} \leftarrow x$, set $\text{alt_local} \leftarrow x$
 obs.: $\text{alt} = x$ implies in $\overline{\text{alt}} = y$ and vice versa
 $\text{ql}(\&\text{alt})$ means queue length of through traffic on the
 queue pointed by alt .

```

at frame tick do
  case of RT(packet_destination)
    arrival:
      buffer it local_in buffer
      mark frame as free
    change_direction:
      buffer it queue(& $\overline{\text{alt}}$ )
      mark frame as free
  esac
  if counter  $\neq$  0 and no frame requested then
    counter  $\leftarrow$  counter - 1
  if frame requested then
    release free frame
  else if alt = alt_local and counter = 0 and local packet ready then
    send local
    counter  $\leftarrow$  C
    alt_local  $\leftarrow$   $\overline{\text{alt\_local}}$ 

```

```

else if ql(&alt) > 0 then
    send from queue(&alt)
else if ql(&alt) > 0 then
    send from queue(&alt)
else if local packet ready then
    send local
    counter ← C
    alt_local ← alt_local
fi
if ql(&alt) > 1 then
    request free frame(alt)
fi
if ql(&alt) > 2 then
    request free frame(alt)
fi
alt ← alt
od

```

The above protocol may provide unfair service under certain conditions. For example, the protocol is biased towards providing half of the frames for each of the incoming links; therefore, a switch could use half of the frames for packets that entered the network on a switch just one hop away (one of the links), and use the other half of the frames for the other link for packets which could have been longer in the network (i.e. that have been already through more hops). One proposed solution is to add one more field in the link, which keeps track of the number of waits that the packet has been through since it was inserted in the network. The packet with the higher wait value is serviced. This was simulated and the results are similar for uniform traffic and present fair allocation of frames for non uniform traffic.

5.3.4 Results

The results we present below are simulations from RESQ models of the GridNet, GridNet/p2 ($FC = 4$), and MRing/NR-II ($C = 16$) with 16 switches each. Figure 5.30 and 5.31 compares delay and throughput versus packet interarrival time.

The two following graphs (Figures 5.32 and 5.33) show delay and throughput versus interarrival time for 16 switch models of the MRing, Dring, and Cubic. In all cases, $C = 16$. Here we also compare the results with the theoretical limit as described earlier for the GridNet case (dotted lines headed with a letter l).

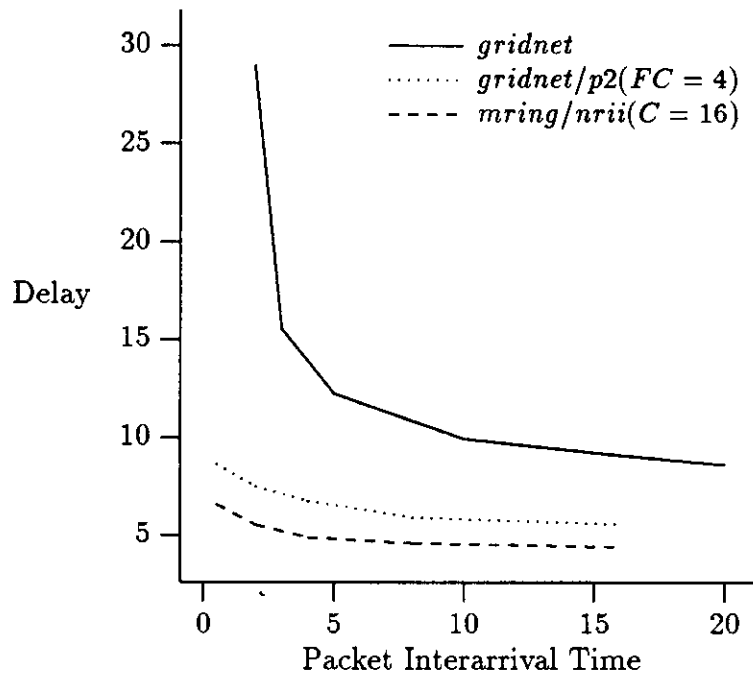


Figure 5.30: Delay versus Packet Interarrival Time

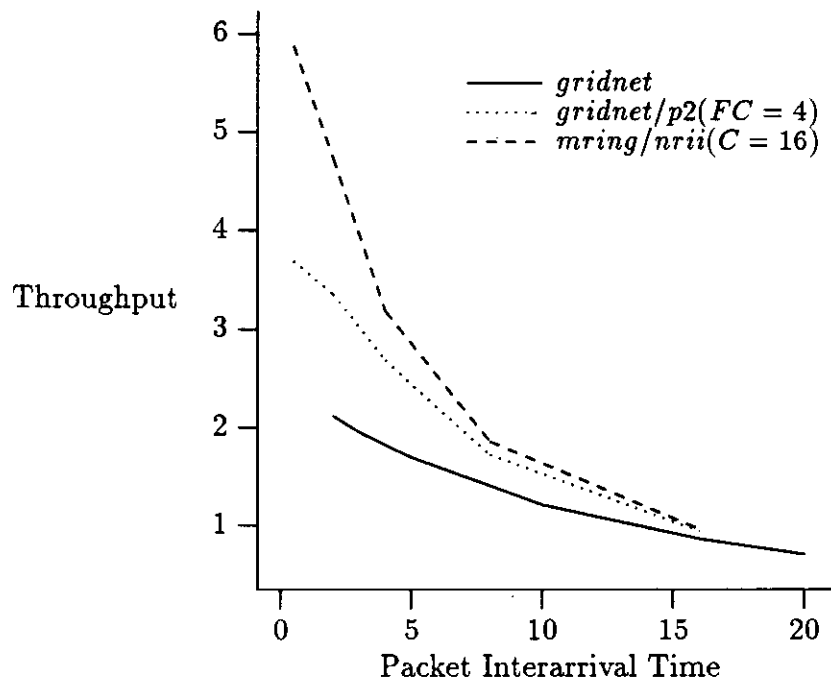


Figure 5.31: Throughput versus Packet Interarrival Time

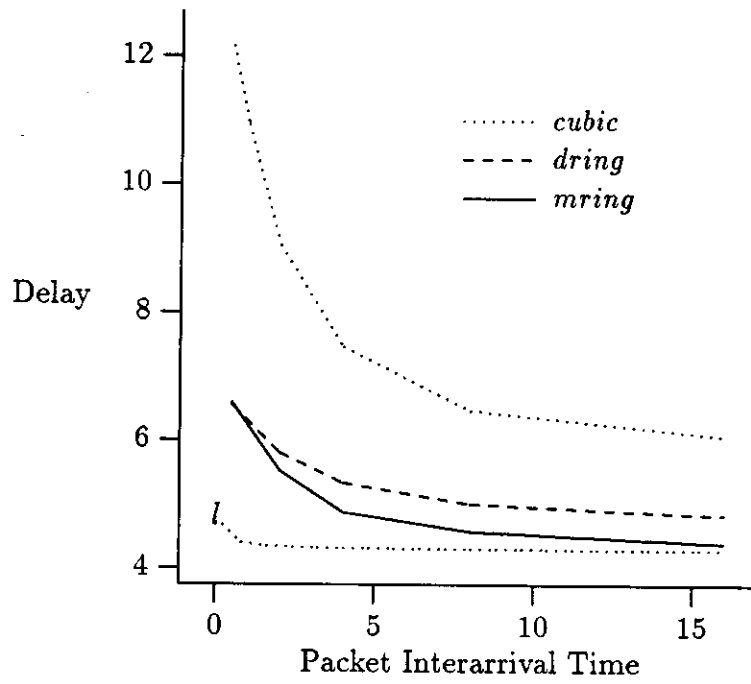


Figure 5.32: Delay versus Packet Interarrival Time

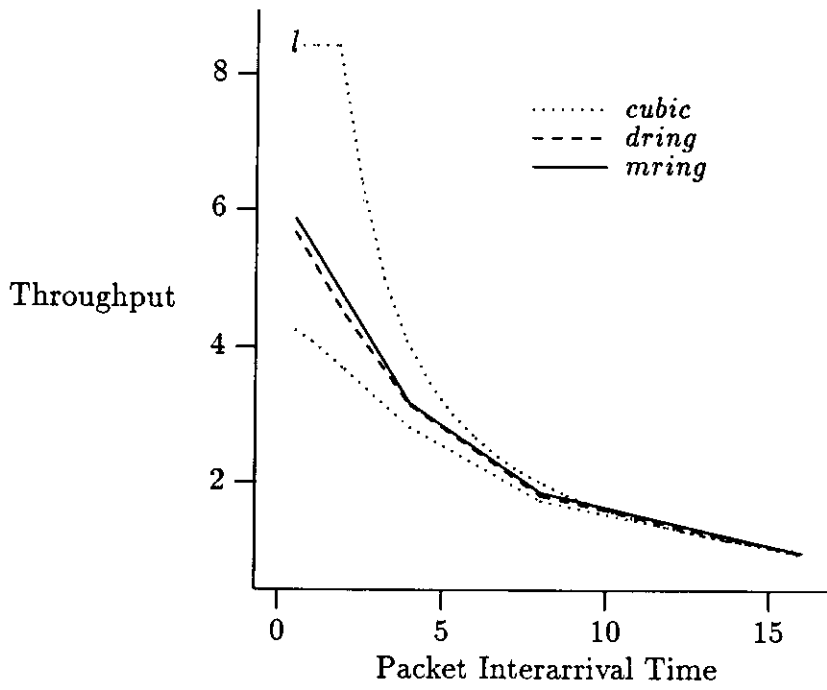


Figure 5.33: Throughput versus Packet Interarrival Time

And the last set (Figures 5.34 and 5.35) compares both protocols, NR-I and NR-II on the Mring topology. Same assumptions as before: $C = 16$, uniform destination, and poisson arrivals. As we can observe, they both have, for these assumptions, the same performance. It was also noted that increasing the usage of buffers at the switches by means of relaxing the threshold when back pressure is applied improves the performance on high load, but not significantly, e.g., 5% for the delay time and 2% for the throughput.

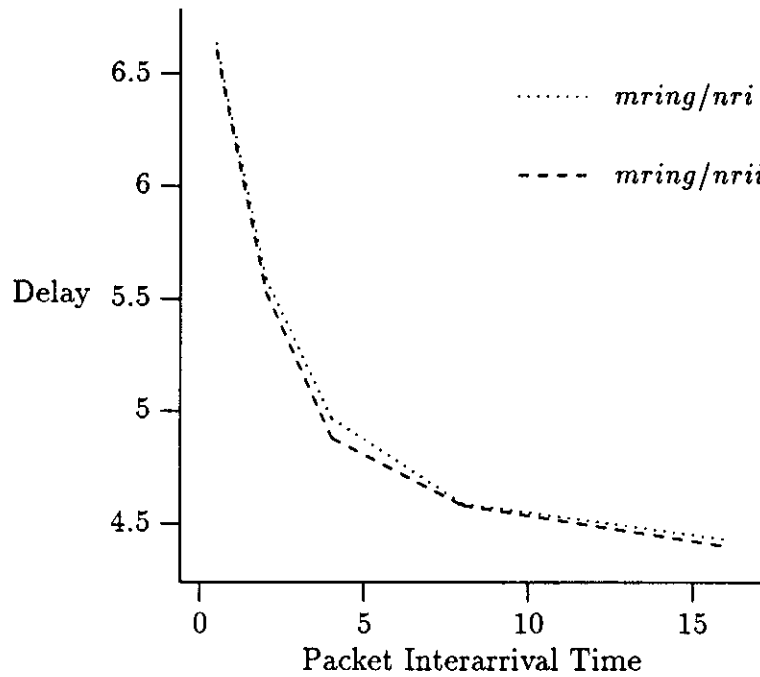


Figure 5.34: Delay versus Packet Interarrival Time

5.4 Conclusions

In this chapter we presented different topologies and protocols aimed at future integrated circuit technologies. We have based our choices of network on slotted single access links rather than multiple access bus. We see the benefit of this choice in two ways: 1) higher throughput because of pipelining (i.e. each frame might be filled) and, 2) reduction in propagation delay due to the limited distance over which the signal travels between two switches.

Our intent was to give an insight into the behavior of various protocols, which in conjunction with the choice of topology, determine the performance of the system. We also looked into the issue of tuning input flow control so to fairly assign bandwidth to different switches.

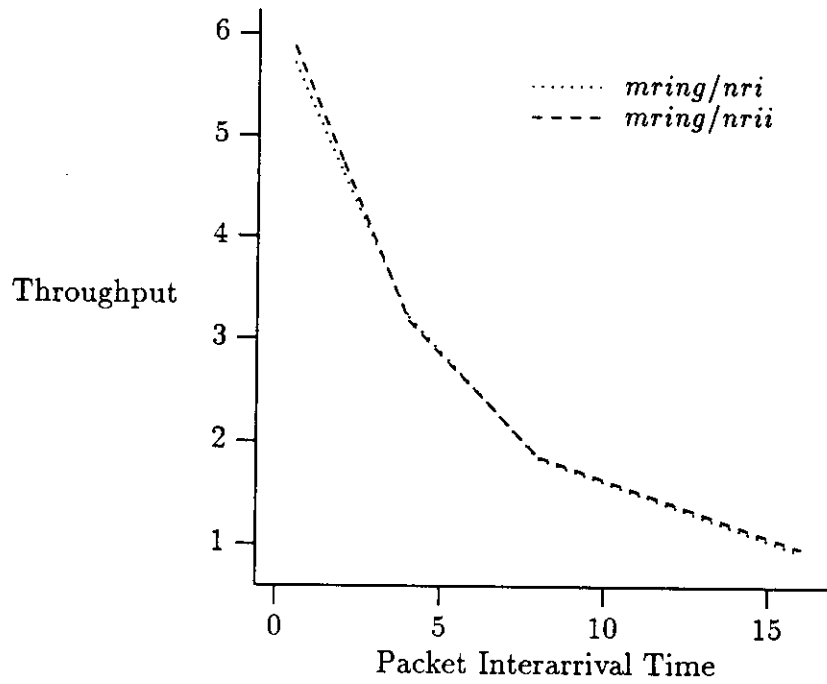


Figure 5.35: Throughput versus Packet Interarrival Time

We believe that the added complexity of the network will be offset by the benefits that this concept brings, namely, easier communications and higher flexibility in chip design.

Chapter 6

Routing

Routing is the function that takes care of moving the packets from source to destination. Routing in a typical network involves some complex algorithms exchanging information. It requires the coordination of all nodes of the subnetwork; it must cope with topological changes in the subnetwork, due to link and/or node failures or repairs, addition of link and/or node and it must also cope with the load conditions, avoiding saturation or congestion. Basically the goals of routing are: selecting routes to achieve high performance and adapting to topological changes. It has been studied in detail for a long time and there is a long list of literature available. Some references are [BG87, Sch87]

6.1 Our Domain

In our particular domain, the routing function has to be performed very fast. Our goal is to achieve packet delay from source to destination in the same order of magnitude than the delay of signals on wires of length equivalent to the distance of modules as seen before. Consequently, there is no time to do any intricate processing while packets are coming into the nodes. Hence, the complex processing of routing is to be done off line or in the background and loaded into the nodes as load conditions change.

We will focus on a topology which is very regular and was detailed in Chapter 5. The network is composed by a small number of nodes (4×4 or 16×16), which are linked by parallel bus wide enough to carry data and control signals. The number of nodes, as explained before, is limited by the tradeoffs between the network size and module (block) size. We also assume that changes in load patterns as well as changes in topology (due to failures) are infrequent when compared with the block

transmission rates.

At each switch, a packet may enter the subnetwork, leave, continue to the next switch in the same direction, or change direction (or plane). The actions related to these events are under the control of the routing function. The network topology can be seen in Figure 6.1.

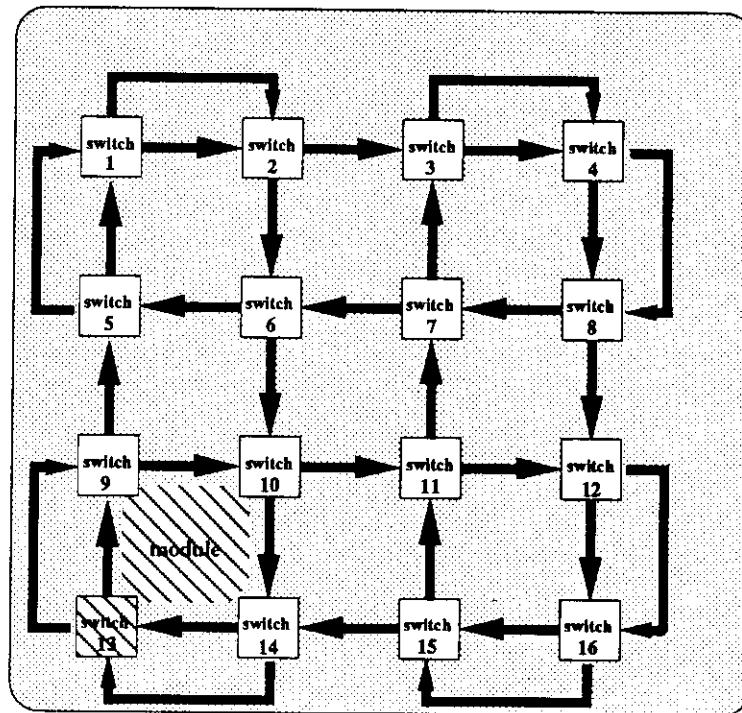


Figure 6.1: Network Topology

6.2 Routing Mechanisms

In order to perform the routing functions in a speedy way, two approaches have been sought. Adaptive routing based on deflection and fixed routing.

The basic idea of deflection is to make the routing decisions at the local level based on immediate conflicts. This idea was first presented by Maxemchuk [Max85, Max87] and further analyzed in [GG86]. Each switch has $2in - 2out$ links and at each time frame, the routing decision is made based on the destination of the

incoming packets. The point being that there is no buffering, therefore, output link conflicts are solved by “deflecting” one of the packets (i.e. one of the packets takes the output link which is not in the shortest path to the destination).

In Figure 6.2 we present the basic architecture of the switch. The routing table is constructed based on the shortest path from this switch to all others. Each entry contains the preferred output link for the given destination. The possible outcomes are: \rightarrow , the preferred output is link X ; \downarrow , the preferred output is link Y ; DC (Don’t Care), either output link is in the shortest path.

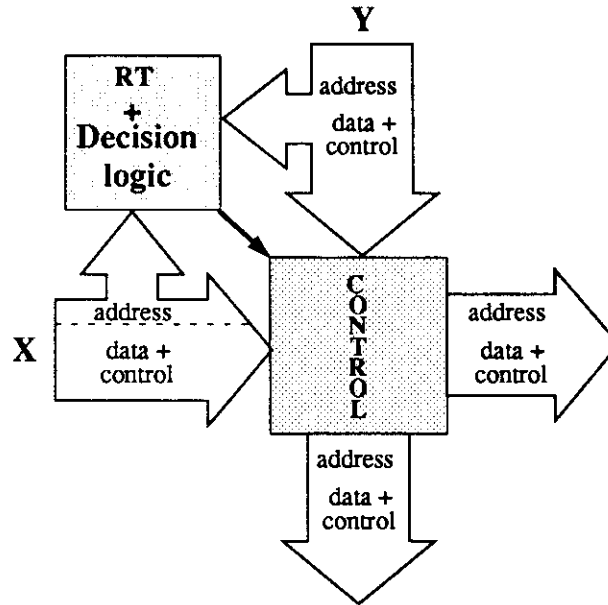


Figure 6.2: Deflection Routing

The final routing decision is made by a logic table that is presented in Table 6.2. The results are: \rightarrow , packet from X exits on X and packet from Y exits on Y ; and \times , when X goes to Y and vice versa.

The idea that no buffering is needed because of the $2in - 2out$ topology is not necessarily true due to the fact that there is also the local access. Maxemchuk [Max85] says that local source is to transmit only when one of the outgoing links is not used by one of the incoming links. However, since fairness is an important issue, the local traffic cannot depend on free frames that eventually might come by.

The fact is that there are 3 inputs and 3 outputs, the third input is used by the local traffic and the third output is used by packets that reach their destination. In our case, since deflection guarantees that there is no back log among the two major links, we may use any of the protocols PI-III to the conflict resolution or

Preferred direction traffic from X	Preferred direction traffic from Y	Result	Observation
→	↓	⊕	y deflected
→	→	⊕	
→	DC	⊕	
↓	↓	⊕	x deflected
↓	→	⊗	
↓	DC	⊗	
DC	↓	⊕	
DC	→	⊗	
DC	DC	⊕	

Table 6.1: Decision Logic for Output Links

contention between the local traffic and the network traffic, which now becomes a decision between one of the links and the input traffic. And buffering, though limited, is needed.

For the fixed routing, two alternative mechanisms have been proposed. The first one (RT) is based on table-lookup by the destination (destination routing) at each switch and the second one (source routing) is based on shift-decide (SD) at each switch. In the first solution, at each switch, the packet destination address is used to access the routing table, which will be used by the switch to take the appropriate action. Therefore, the address on the packet needs $\log_2 N$ lines of the bus; N being the size of the network.

In the second scheme, the path is coded in the packet path field, (which is equivalent to the destination field in the other scheme) using $M + 1$ lines of the bus, where M is the longest path from any source to any destination. In the 4x4 network, $M = 8$. Initially the address field is loaded with the "path" to the destination on the top most bits and the rest is loaded with "0...01." At each switch, the address is shifted out to a single bit register and a "0" is shifted in. The single bit information is used to make the routing decision. When the all-zero address detector is true, the packet has arrived to its destination.

Figure 6.3 and 6.4 illustrate the two schemes. The differences among these schemes in a 4 by 4 network are:

- **Memory requirement:** The RT scheme requires at each switch 3 tables with 16 entries of 2 bits, totaling 1536 bits. Two tables are for the decision in the different planes and the other one is used for the local traffic. The SD scheme requires a total of 2160 bits, stored in the modules (blocks), each one having 15 entries (one for each destination) of 9 bits each (the longest shortest path +1).

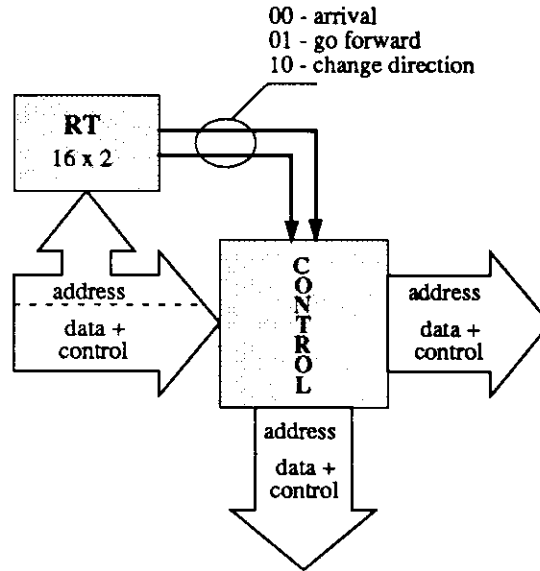


Figure 6.3: Routing with Table-Lookup (RT)

- Complexity involved: About the same complexity. The RT needs a fast decoder for the table entries, while the SD needs a fast zero detector for bits across the address bus. The control and buffer management can be the same.
- Length of address/path field: $\log_2 N$ lines are needed in the RT scheme. In the SD scheme, $O(2N)$ lines are needed, where $O(2N)$ is the longest path used + 1.
- Routing dependencies: Routing decisions in the RT scheme are based on the destination of the packet (destination driven) and this puts an additional constraint on the possible path for packets. If $\text{Path}\{S_1 \rightarrow D\}$ goes through switch V and $\text{Path}\{S_2 \rightarrow D\}$ also goes through switch V , then both routes share $\text{Path}\{V \rightarrow D\}$, since the decision in V is only based on D . In the SD scheme, routing is independent, since it is self contained.
- Update or change in routing tables: In the RT scheme, the routing table is kept in the switch. Updates have to be downloaded to the switches. In the SD scheme, the address mapping should primarily be kept in the module (block), since the switch has no use for the destination mapping mechanism to the path itself, which is what it uses.

In order to compare the deflection routing and the fixed routing, a study was made based on simulation of a 4x4 network (see Figure 6.1). The fixed routing uses RT alternative. The results can be seen in Figure 6.5. The assumptions are uniform destination and exponential arrivals. As expected, (see Theorem 4)

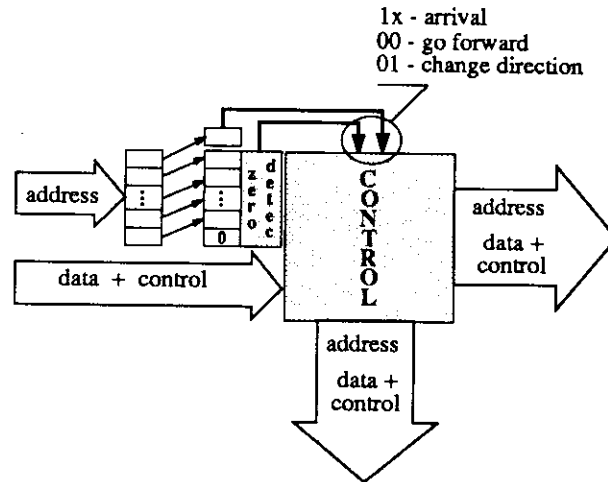


Figure 6.4: Routing with Shift and Decide (SD)

the deflection routing has a lower performance than the fixed routing because of longer path that packets might take. When this happens, the packet will “stay” longer in the network, and occupy frames that otherwise could be used by other packets. Two other drawbacks of deflection routing are that delay is not bounded and packets might arrive out of order.

6.2.1 Broadcasting

The routing table mechanism with some additional logic can also provide broadcasting of packets. A new signal on the links is needed to indicate whether the frame contains a broadcast packet or not.

At the switches, an additional RT is used to make the routing decision of the broadcast packets. The possible outcomes are: send out on X , send out on Y , send out on both (X and Y), and DE (dead end). Table 6.2.1 exemplifies the RT entry in all switches for a broadcast originating from $switch_{10}$. The links used in this broadcast are shown in Figure 6.6. Broadcast packets “arrive” to all switches that they pass through.

The originating switch, uses its own address on the packet address, unlike the normal packets that use the destination address. The basic idea is that the corresponding routing table entries implement a minimum spanning tree from the originating switch to all others switches.

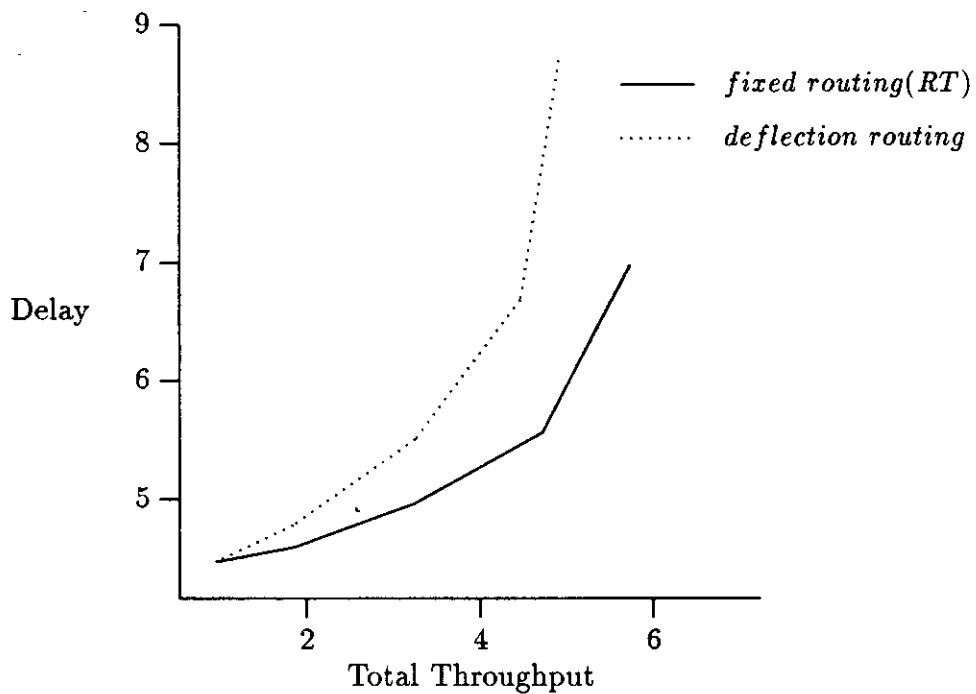


Figure 6.5: Comparison Between Fixed and Deflection Routing

Switch	Entry for RT(10)	Switch	Entry for RT(10)
1	<i>X</i>	9	<i>DE</i>
2	<i>DE</i>	10	<i>XY</i>
3	<i>X</i>	11	<i>XY</i>
4	<i>Y</i>	12	<i>X</i>
5	<i>Y</i>	13	<i>Y</i>
6	<i>X</i>	14	<i>X</i>
7	<i>XY</i>	15	<i>DE</i>
8	<i>DE</i>	16	<i>X</i>

Table 6.2: Table Entries for Broadcast Originating from *switch*₁₀

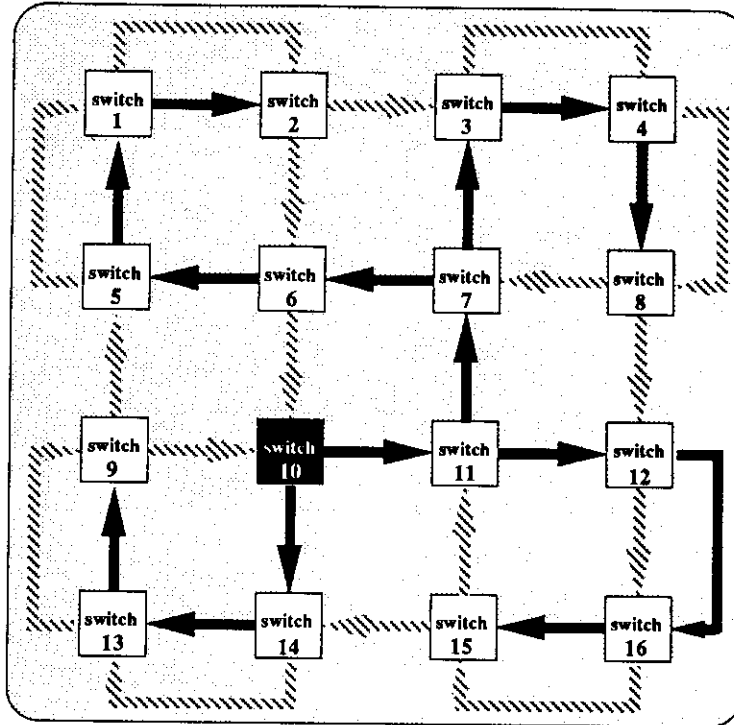


Figure 6.6: Broadcast Originating from $switch_{10}$

6.3 The Routing Problem

From the section above we saw the mechanism through which messages can be routed from origin to destination within the subnetwork. We are now going to address the problem of creating those tables, when an optimal solution is sought regarding link utilization, or minimum time, for example. The problem is that there are too many paths, and we are already constraining the problem to min-path, e.g., only routes with the shortest path are considered. Actually, there is a good reason to restrict ourselves to min-path, besides the obvious reason to limit our search space, as we will see in the following theorem.

Theorem 4 *In a mesh with unidirectional links, where directions alternate horizontally and vertically, if $Path\{S \rightarrow D\}$ is not a shortest path, then the number of hops from S to D is $\geq 4 +$ number of hops in the shortest path from S to D .*

Suppose $S \dots J \dots D$ is the shortest path. If we “avoid” the link that leads us to $switch_j$, and take the one that take us to K , there are two possibilities:

1. $S \dots K \dots D$ is also a shortest path, however, we are not interested in this case.

2. $S \cdots K \cdots D$ is not in the shortest path, it means that we are moving away by one hop. At the next *switch*, either path we take, we will also be moving away:

- same direction - if in the previous step we moved away, we are also moving away in this step.
- change direction - since we took the path to K , changing direction again will mean moving in the opposite direction than J , hence we are also moving away.

At this point where we moved 2 steps further from the sought destination, we can start moving back in the direction of D , but it will take another 2 steps.

For example, the shortest path from 6 to 14 is:

$6 \rightarrow 10 \rightarrow 14$: (2hops)

if at node 10 we decide to take a different direction, the new path will be:

$6 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 15 \rightarrow 14$: (6hops)

Corollary 1 *The number of hops from S to D is $4 \cdot k + d$, where d is the number of hops for the shortest path and k is an integer including 0.*

Since the networks in our study are small, adding 4 hops to a path, would increase the time for packets routed that way by a large margin, even when this is done to avoid congested links. Two reasons being: 1) the traffic would be diverted to a longer path and 2) would contribute to the traffic each of the added path. For instance, in the small networks we are considering (4x4) the longest shortest path is 8.

6.4 Routing Algorithm

For the 4x4 network with all nodes communicating with each other (i.e. 15x16 pairs), the total number of possible minpaths is 1884. The solution to the optimal routing problem is to select out of the total number of min-paths, 16x16=240 paths that satisfy all source-destination pairs and have the minimum load on the links for a given load/connectivity condition. It must also satisfy the routing mechanism constraints (we have chosen the RT mechanism in this study). The following table (Table 6.3) summarizes the distribution of alternatives.

number of source destination pairs	number of alternatives pairs
48	1
56	2
52	4
12	5
16	8
16	10
8	16
20	20
8	40
4	80

Table 6.3: Distribution of Alternative Minpaths Between Source-Destination Pairs

The goal of optimizing the routing tables so as to spread the traffic among the links as much as possible, reduces the accumulation of messages in the links, avoiding bottlenecks, and therefore increasing the overall throughput. Of course there are other optimization criteria, however this choice makes a fair use of the links and minimizes the average time in a uniform load assumption.

The problem of generating the optimal routing tables is a formidable one. The search for the optimal (or a reasonable suboptimal) solution should not be based on trying out all the different combinations of routing table, it would take *too long* for the available resources.

Hence, we propose a heuristic algorithm whose goal is to spread the load on the links as evenly as possible. The first step of the algorithm is to find an initial feasible (i.e. consistent) set of routing tables. Then, repeatedly we reduce the load on the most utilized link, until no improvement can be made. The load reductions is accomplished by selecting a communicating pair which uses this link and reroute it through a different path such that it lowers the “spread” (standard deviation on the link utilization).

Since the problem is an integer problem, e.g., flow from an origin to a destination cannot be split on multiple paths (non bifurcated flow) there is the “danger” of getting caught on a local minimum instead of the global minimum. At this point, there is no traffic that can be deviated from this link that will improve the objective function. We solve this by “perturbing” the solution: we choose a link at random and apply the minimization process.

The algorithm goes as:

```

find a feasible/consistent RT
while global_improvement > Epsg do

```



```

while local_improvement > Epsl do
  find Lmax
  findpairs(Lmax)
  for all pairs do
    find altpath(pair)
    elimpath_from_rt(Lmax)
    for all path do
      try_alt(pair)
      if feasible/consistent then
        if objective(load) improved then
          save_result
        fi
      fi
    od
  od
  if improvement then
    update rt
  fi
od
perturb
od

```

6.5 Applying the Results

The resulting RT solution was tested via simulation, and showed great improvement over the initial RT choice as shown in Figure 6.7 and 6.8. As expected, the improvement is particular significant at high offered loads.

When a packet switches direction, it incurs a latency delay due to the asynchrony of the frames on the two orthogonal planes. Therefore, if we restrict our choice of the possible paths to the ones that have minimum of direction changes, we should expect a better performance under a uniform load assumption.

In this restricted case, the total number of paths is 360 (as compared with 1884 in the non restricted case). The solution therefore, is to choose the best 240 source-destination pairs out of this 360.

Applying the algorithm described above with this restriction, and evaluating the resulting RT via simulation, the expected improvement was observed and is seen as dashed lines in Figures 6.7 and 6.8. The benefit of the optimization is even more evident on the *Power* function, which is shown in Figure 6.9.

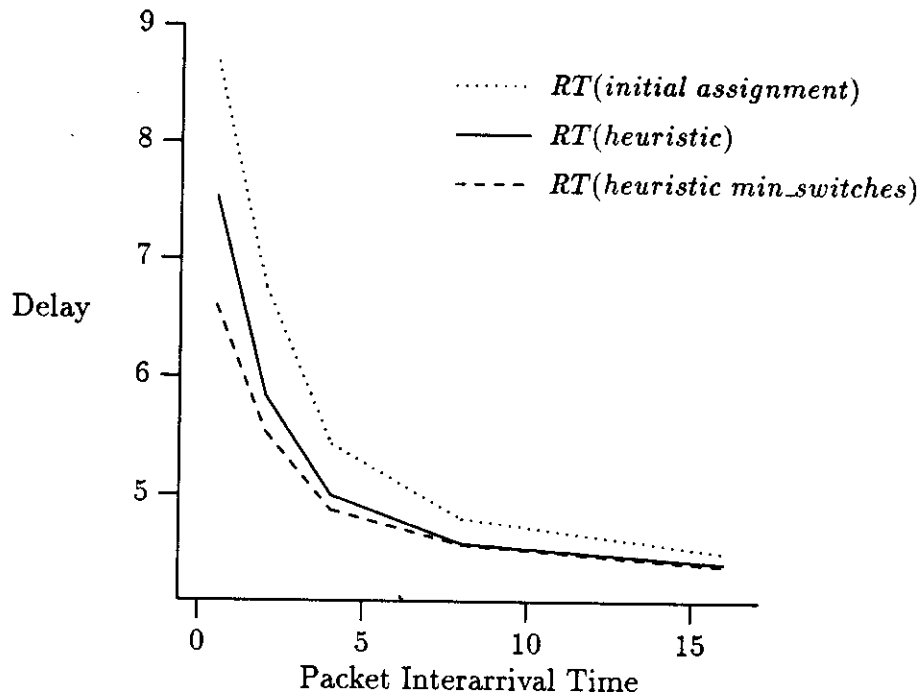


Figure 6.7: Total Delay versus Interarrival Time for Initial and Optimal RT

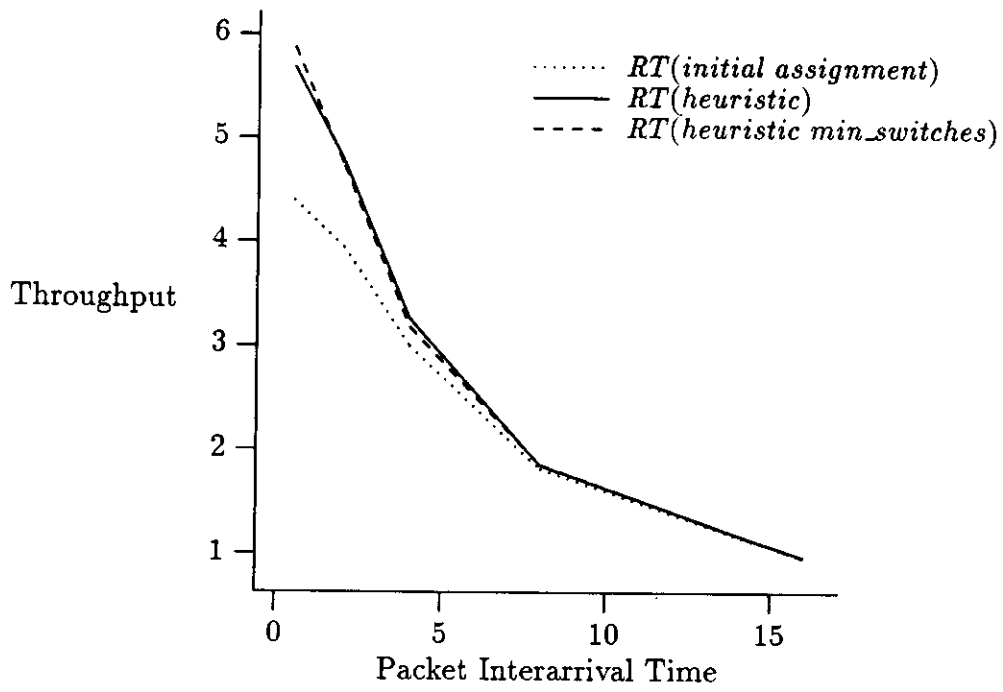


Figure 6.8: Total Throughput versus Interarrival Time for Initial and Optimal RT

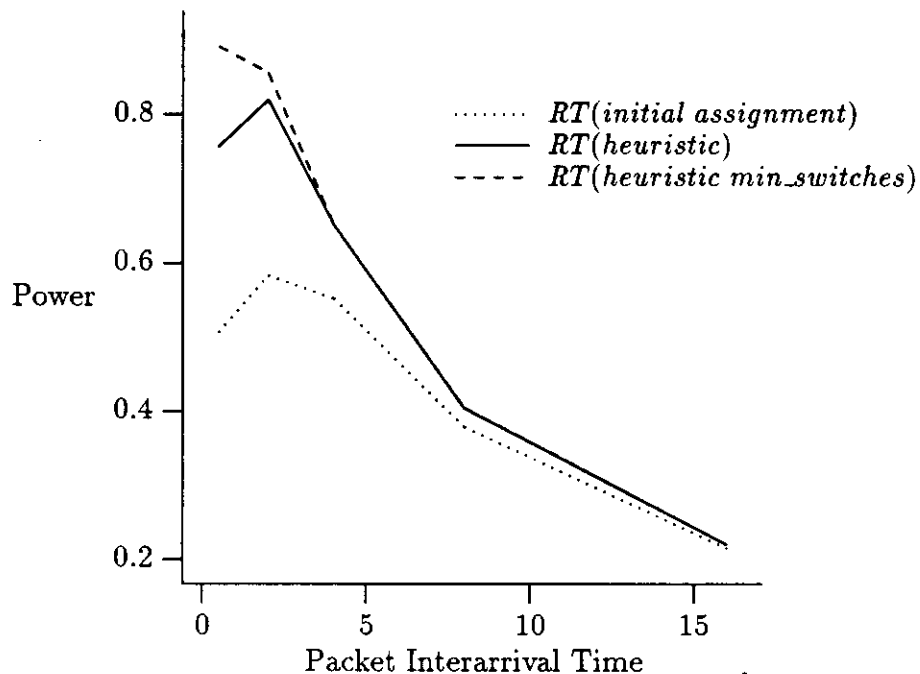


Figure 6.9: Power versus Interarrival Time

6.6 Routing Assignment by Flow Deviation

In order to obtain a lower bound on the problem (and thus estimate the accuracy of the heuristic algorithm), we implemented a flow-deviation algorithm based on [FGK73].

In this case, we assume that our problem is a multicommodity flow problem (MFC) consisting in minimizing a delay function such that a set of constraints (link capacities) are satisfied. Formally it can be expressed as follows:

Given: Topology (Figure 6.1), channel capacities (one packet per unit time), and requirement matrix R (load requirement)

$$\text{Minimize: } T(\underline{f}) = \sum_{i=1}^b \frac{f_i}{\gamma} \left[1 + \frac{f_i}{2(1-f_i)} \right] \text{ over } \underline{f}$$

Constraints: (i) f is a multi commodity flow
(ii) $f_i \leq 1$, for $i = 1, \dots, b$

The function $T(\underline{f})$ corresponds to the average delay from source to destination over time and over all pair of nodes.

An important consideration is that we are assuming that this network is a

Product Form Network [Jac57, BCMP75] so that we can apply the expression above. γ is the total arrival rate of local packets to the network.

The expression $1 + \frac{f_i}{2(1-f_i)}$ corresponds to the queueing delay of a $M/D/1$ queueing system [Kle75] and corresponds to the deterministic service of the queues (nodes in our case) with service time equal to 1 (our unit time) and an arrival rate of f_i (assuming poisson arrivals).

Let F_a be the set of feasible flows: $F_a = F \cap \{\underline{f} \mid \underline{f} \leq \underline{1}\}$.

Clearly F_a is a convex set [FGK73]. Therefore a minimum $T(\underline{f})$ can be found by the Flow Deviation Method.

The algorithm goes as follows:

```

while (improvement >  $\epsilon$ ) do
    find flow through links for this instance of RT
    find minimum cost path
    create the alternate RT corresponding to min cost paths
    find flow through links for alternate RT
    find the optimal balance between these two flows ( $\theta$ )
    update RT using  $\theta$ 
    compute improvement
od

```

From the resulting flow, we have a RT that has optimal flow spreading characteristic.

It is important to understand that by assuming our problem to be a MFC, we are relaxing some of the constraints of the original problem. Flow can now be bifurcated at any switch, therefore, it is not destination driven any more.

6.7 Comparing the Algorithms

We compare the H algorithm (heuristic algorithm) and FD (flow deviation) algorithm presented under two different load conditions: 1) uniform and 2) non uniform.

For the non uniform load, a $node_i$, chooses a destination based on:

$$destination_i = Random_i\{1, 16\} \cdot \phi + Fix_i\{1, 16\} \cdot (1 - \phi)$$

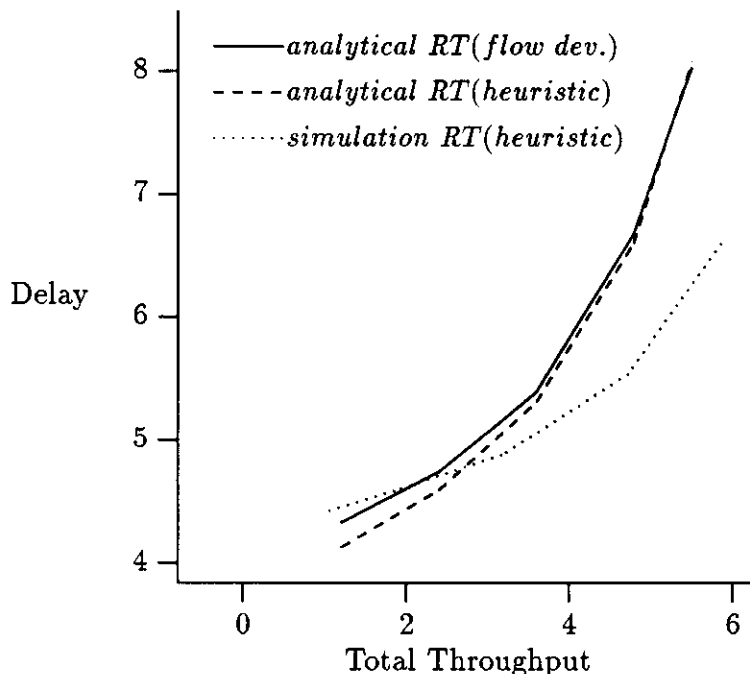


Figure 6.10: Total Time versus Total Throughput for Uniform Load

where, $Random_i$ returns an integer random value uniformly distributed between 1 and 16, excluding i . Fix_i returns a fixed integer value arbitrarily chosen at the beginning of the computation which excludes i . ϕ determines the percentage of traffic that is uniform. In our study, $\phi = 0.7$.

The comparison of the “optimal” RT generated by the heuristic algorithm and the one generated by the FD algorithm presented close results for the uniform load condition (Figure 6.10). The RT(flow dev.) behaves slightly better for heavy load, as expected because of the traffic bifurcation, however, the difference is really small (solid line for RT(flow dev) and dashed line for RT(heuristic)). The difference at low throughput between analytical RT(flow dev.) and analytical RT(heuristic) is explained by the lack of sensitivity of the FD algorithms at low load (mainly due to the stopping conditions). Also in the figure, we show the performance for the simulation result (dotted curve).

At low load, the simulation delay is higher than the analytical RT(heuristic) because arriving packets in the simulation have to wait for the frame tick, which is not accounted for in the analytical model. This wait is the synchronization time t_{sync} .

The synchronization time t_{sync} is $1 - \bar{t}$ and \bar{t} is the average arrival time within the frame period, normalized by the probability of an arrival in a frame period. The assumption on the local buffer being of size one (see Chapter 5) limits the

number of arrivals to one within one frame period. Assuming poisson arrivals, which enjoy the memoryless property and frame period of one, we have:

$$t_{sync} = 1 - \frac{\int_0^1 t \lambda e^{-\lambda t} dt}{\int_0^1 \lambda e^{-\lambda t} dt}$$

which results,

$$t_{sync} = \frac{1/\lambda - e^{-\lambda}(1 + 1/\lambda)}{1 - e^{-\lambda}}$$

At higher load, the simulation performance result is better because in the simulation the waiting room for local arrivals is one; therefore the network does not get as congested as the one represented by the analytical model. The simulation was designed with limited waiting room since we assume a handshake protocol between the module and the switch that allows for the module to transmit only place one packet at a time. The analytical model assumes no such limitation.

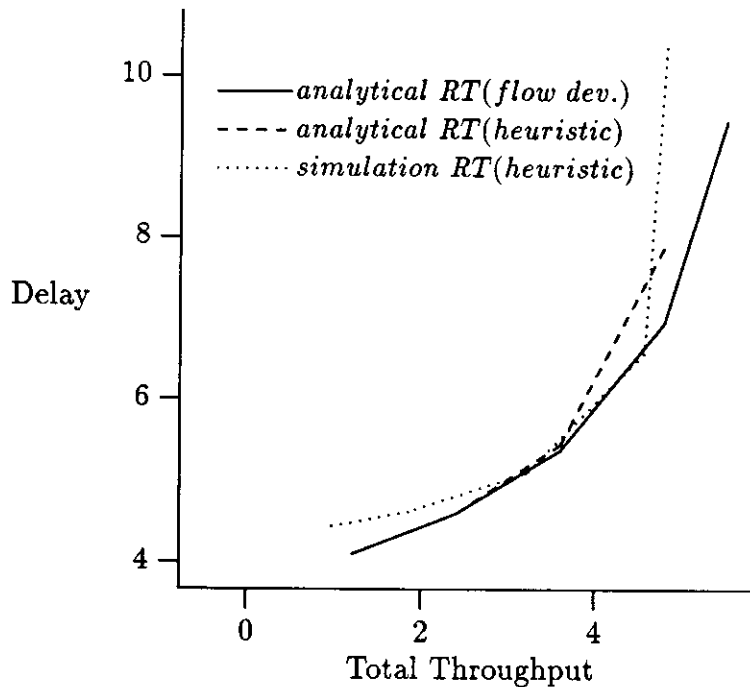


Figure 6.11: Total Time versus Total Throughput for Non Uniform Load

In the non uniform experiment (Figure 6.11), it is interesting to note that the analytical RT(flow dev.) performed better than the analytical RT(heuristic) due to bifurcation. This allowed the traffic to “spread” over many links and therefore relieve the congested ones.

6.8 Conclusions

In this chapter, we have studied different routing mechanisms from which we concluded that in general, fixed routing is more suitable for the networks we are interested in.

We have also proposed an heuristic algorithm for routing table generation and have compare the results with optimal solutions obtained with the Flow Deviation algorithm as well as with simulation. We have found that the heuristic solution is very close to the optimum.

Chapter 7

Conclusions

7.1 Summary of Results

The development of the VLSI technology has been very beneficial to the computer field by achieving greater integration and faster circuits. On the other hand, this development uncovers challenging problems. One of these problems is the long distance interconnection on chip which is increasingly becoming the dominant factor in the performance MOS technology integrated circuit, as well as in chip real estate (area consumption).

This work proposes new communication mechanisms that achieve cost effective solutions to overcome the high delay of signals over long wires. The problem of area usage is minimized by sharing the communication resources.

Furthermore, the idea of a network on chip is very attractive since it helps in dealing with the complexity of VLSI design by enforcing the hierarchical methodology approach. It also defines a common interface by which modules communicate. This makes easier the design effort of a complex system. Sharing of the resources on chip is likewise facilitated by the network as well as testing, maintenance, and fault isolation. Moreover, area is saved because of the sharing of the communication channels.

In Chapter 2, we presented the basic components of VLSI interconnection, and discussed the effects of scaling on the characteristic of the basic components. We also introduced a delay model for the line/driver pair, which was used to evaluate different driving schemes for long distance interconnection. Furthermore, we presented the beneficial impact of pipelining in terms of delay and throughput for an efficient driving schemes.

In Chapter 3, we did a literature review in the search for networks that would fit our constraints. This gave us the necessary background material to discuss different topologies and other design issues under the VLSI framework.

In Chapter 4, we presented our understanding of the requirements for networking on chip with discussion on the advantages and details. A comparison of this network with more traditional ones was also included.

In Chapter 5, we presented and discussed different topologies and protocols that would suit the VLSI environment. Many different aspects were studied.

In Chapter 6, we looked into routing schemes and the routing problem for which we developed an algorithm based on heuristics which achieved good results when compared with optimal solutions.

Our major contribution was a comprehensive examination of protocols for the proposed VLSI networks (GridNet and RingNet) presenting issues like liveness, flow control, fairness, and buffer needs. Performance results based on simulation were also discussed.

An important and interesting finding was that delay could be lowered and at the same time throughput could be increased on a long distance interconnection by segmenting the line and buffering the signal at these points. This implies that the preferred network solution for VLSI is slotted.

Another contribution was the development of an heuristic algorithm to generate the routing tables that are used in the networks.

7.2 Extensions of this Work

There is much work to be done since this is a relatively untouched area.

Our research concentrated mostly in the conceptual idea of the network on chip. An important next step is to look further in the implementation details of such networks in order to uncover issues that have not been dealt with yet and propose alternatives to circumvent them. Among them are:

- Specification of the interface between the switch and module in terms of required signals, their timing, and handshaking protocol;
- Details of buffer management (for instance, first in first out strategy using fast registers with simple control, since the buffer requirement is quite small);
- Implementation of fast access routing tables;

- Viability study of overlapping internal switch process functions, such as table lookup and routing decision (if this can be achieved, the latency of the switches may be reduced);
- Analysis of the tradeoffs between area dedicated to the network and area available for the processing modules (e.g. by increasing the spacing between switches, we may increase the VLSI chip processing power, but we may decrease the network throughput).

Another interesting aspect is the development of analytical models to study larger networks since we are close to the limits of simulation. In our experiments, each simulation run took many hours of cpu time, making the testing of larger networks impractical. There is also to be learned from a more detailed study of these networks under different load conditions. However an analytical approach (usually a difficult endeavor) cannot incorporate as much detail as the simulation otherwise it easily becomes intractable. Decomposition could be used, where protocols would be modeled in isolation and later these results applied to the network model.

The concept of networking gives a different perspective to the issues of reliability and fault tolerant besides testability. The separation of the processing/server modules and the communication among them with a defined and common interface can make reconfiguration simpler. Analysis on the effect of losing links, can be made based on the routing algorithm developed for this work to find out the performance impact as well as how modules are partitioned in terms of reachability.

Most of the proposed protocols in this work do flow control on a line basis. This in certain traffic conditions might impact nodes that are not contributing to the congestion that triggered the flow control. One possible solution would be to perform "input" flow control based on destination. This would also require more logic on the switches, besides more buffers for the local traffic. The extension of this work, lies in the study the cost and the benefits of such approach.

A natural next step for the continuation of this work would be to analyze this network within a larger context like Wafer Scale Integration and all related issues of optimal interconnection, protocol, flow control, routing. Reliability is one of the major issues in the development WSI and the flexibility of the networking concept might be very interesting in this realm.

The support of real time traffic connection, some kind of "circuit switching" in the context of the GridNet could be of interest for applications/algorithms that require high bandwidth. Here we also note that fast switching fabrics will be required for the upcoming B-ISDN (Broadband Integrated Services Digital Networks) and the fast and streamlined protocols could be of use.

Finally, also of interest is the extension of this work to include optical compo-

nents and optical interconnection. This is becoming a very active area of research. The grid architecture should be revised to take into account the different characteristics of the optical switching elements, optical buffers and optical (waveguide or free space) transmission links.

Bibliography

- [Anc82] Francois Anceau. A synchronous approach for clocking vlsi systems. *IEEE Journal of Solid State Circuits*, SC-17(1):51-56, February 1982.
- [BCMP75] F. Baskett, K. M. Chandy, R. R. Muntz, and F. Palacios. Open, closed and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22:248-260, 1975.
- [BG87] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, 1987.
- [BJJF84] Robert J. Bayruns, R. L. Johnston, Donald L. Fraser Jr., and San-Chin Fang. Delay analysis of nmos gbit/s logic circuits. *IEEE Journal of Solid State Circuits*, SC-19(5):755-764, October 1984.
- [BK83] Melvin A. Breuer and Anshul Kumar. A methodology for custom vlsi layout. *IEEE Transaction on Circuits and Systems*, CAS-30(6):358-364, June 1983.
- [BM84] H. Bakoglu and J. Meindl. Optimal interconnect circuits for vlsi. In *Proceedings 1984 IEEE International Solid State Circuits*, pages 164-165, February 1984.
- [BM85] H. B. Bakoglu and James D. Meindl. Optimal interconnection circuits for vlsi. *IEEE Transaction on Electron Devices*, ED-32(5):903-909, May 1985.
- [BPP82] Gianfranco Bilardi, Michele Pracchi, and Franco P. Preparata. A critique of network speed in vlsi models of computation. *IEEE Journal of Solid State Circuits*, SC-17(5):696-702, August 1982.
- [CLYP81] P-Y. Chen, D. H. Lawrie, P-C. Yew, and D. A. Padua. Interconnection networks using shuffles. *IEEE Computer Magazine*, 14(12):55-64, December 1981.

- [CPM86] H. C. Card, W. Pries, and R. D. McLeod. Contributions to vlsi computational complexity theory from bounds on current density. *The VLSI Integration*, 4:175–183, 1986.
- [DGY⁺74] R. H. Dennard, F. H. Gaensslen, H. N. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Desing of ion implanted mosfet with very small physical dimensions. *IEEE Journal of Solid State Circuits*, SC-9:256–268, October 1974.
- [FGK73] L. Fratta, M. Gerla, and L Kleinrock. The flow deviation method: An approach to store-and-forward communication network design. *Networks*, (3):97–133, 1973.
- [FK85] Allan L. Fisher and H. T. Kung. Synchronizing large vlsi processor arrays. *IEEE Transactions on Computers*, c-34(8):734–740, August 1985.
- [Fra81] M. A. Franklin. Vlsi performance comparison of banyan and cross-bar communication networks. *IEEE Trasactions on Computers*, C-30(4):283–291, April 1981.
- [FT84] Michael Fine and Fouad A. Tobagi. Demand assignment multiple access schemes in broadcast bus local area networks. *IEEE Transactions on Computers*, c-33(12):1130–1159, December 1984.
- [Fuj83] R. M. Fujimoto. *VLSI Communication Components for Multicomputer Networks*. PhD thesis, Computer Science Division (EECS), University of California, Berkeley, CA 94720, 1983. Published as Technical Report UCB/CSD 83/136.
- [Gai83] H. Richard Gail. *On the Optimization of Computer Network Power*. PhD thesis, Computer Science Department, University of California, Los Angeles, CA 90024, 1983.
- [Gam81] Abbas El Gamal. Two dimensional stochastic model for interconnection in master slice integrated circuits. *IEEE Transanction on Circuits and Systems*, CAS-28(2):127–138, February 1981.
- [GD85] Lance A. Glasser and Daniel W. Dobberpuhl. *The Design and Analysis of VLSI Circuits*. Addison-Wesley, 1985.
- [GG86] Albert G. Greenberg and Jonathan Goodman. Sharp approximate models of adaptive routing in mesh networks. *Teletraffic Analysis and Computer Performance Evaluation*, pages 255–270, 1986.
- [GK80] M. Gerla and L. Kleinrock. Flow control: A comparative study. *IEEE Trasactions on Communications*, COM-28(4):553–574, April 1980.

- [GMS87] Donald S. Gardner, James D. Meindl, and Krishna C. Saraswat. Interconnection and electromigration scaling theory. *IEEE Transaction on Electron Devices*, ED-34(3):633–643, March 1987.
- [HJ87] Nils Hedenstierna and Kjell O. Jeppson. Cmos circuit speed and buffer optimization. *IEEE Transaction on Computer Aided Design*, CAD-6(2):270–281, March 1987.
- [HLSM82] Leonard S. Haynes, Richard L. Lau, Daniel P. Siewiorek, and David W. Mizell. A survey of highly parallel computing. *IEEE Computer Magazine*, 15(1):9–24, January 1982.
- [HMD78] W. R. Heller, W. F. Mikhail, and W. E. Donath. Prediction of wiring space requirements for lsi. *Design Automation and Fault Tolerant Computing*, pages 117–144, 1978.
- [Jac57] J. R. Jackson. Jobshop-like queueing systems. *Operations Research*, 10:131–142, 1957.
- [KB84] Israel Koren and Melvin A. Breuer. On area and yield considerations for fault tolerant vlsi processor arrays. *IEEE Transaction on Computers*, C-33(1):21–27, January 1984.
- [Kle75] Leonard Kleirock. *Queueing Systems, Volume I: Theory*. John Wiley and Sons, 1975.
- [Kun82] H.T. Kung. Why systolic architectures? *IEEE Computer Magazine*, 15(1):37–46, January 1982.
- [MA71] G. E. Marihugh and R. E. Anderson. The *h* diagram: A graphical approach to logic design. *IEEE Transaction on Computers*, pages 1192–1196, 1971.
- [Max85] N. F. Maxemchuk. Regular mesh topologies in local and metropolitan area networks. *AT&T Technical Journal*, pages 1659–1685, September 1985.
- [Max87] N. F. Maxemchuk. Routing in the manhattan street network. *IEEE Transactions on Communications*, COM-35(5):503–512, May 1987.
- [Maz87] Pinaki Mazumder. Evaluation of on-chip static interconnection networks. *IEEE Transactions on Computers*, C-36(3):365–369, March 1987.
- [MBL⁺89] Alain J. Martin, Steven M. Burns, T. K. Lee, Drazen Borkovic, and Pieter J. Hazewindus. The design of an asynchronous microprocessor. In *Decennial Caltech Conference on Advance Research in VLSI*, pages 351–373, 1989.

- [MC80] Carver Mead and Lynn A. Conway. *Introduction to VLSI Systems*. Addison-Wesley Publishing Company, October 1980.
- [MR79] Carver Mead and Martin Rem. Cost and performance fo vlsi computing structures. *IEEE Journal of Solid State Circuits*, SC-14(2):455–462, April 1979.
- [MR82] Carver Mead and Martin Rem. Minimum propagation delays in vlsi. *IEEE Journal of Solid State Circuits*, SC-17(4):773–775, August 1982.
- [MRRS84] Jack F. McDonald, Edwin H. Rogers, Kenneth Rose, and Andrew J. Steckl. The trials of wafer scale integration. *IEEE Spectrum*, 21(10):32–39, October 1984.
- [Muk85] Amar Mukherjee. *Introduction to nMOS and CMOS VLSI Systems Design*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
- [NDN87] Zhen-Qiu Ning, Patrick M. Dewilde, and Fred L. Neerhoff. Capacitance coefficients for vlsi multilevel metallization lines. *IEEE Transactions on Electron Devices*, ED-34:644–649, March 1987.
- [Pie82] J. R. Pierce. Network for block switching of data. *Bell System Technical Journal*, 51:1133–1143, July/August 1982.
- [PV81] Franco P. Preparata and Jean Vuillemin. The cube-connected cycles: A versatile network for parallel computation. *Communications of the ACM*, 24(5):300–308, May 1981.
- [Ram82] Vijaya Ramachandran. On driving many long lines in a vlsi layout. In *Proceedings 23rd Symposium on the Foundations of Computer Science*, pages 369–378, 1982.
- [Rei83] Arnold Reisman. Device, circuit, and technology scaling to micron and submicron dimensions. *Proceedings of the IEEE*, 71(5):550–565, May 1983.
- [Sak83] T. Sakurai. Approximation of wiring delay in mosfet lsi. *IEEE Journal of Solid State Circuits*, SC-18(4):418–426, August 83.
- [SAS83] Charles H. Stapper, Frederick M. Armstrong, and Kiyotaka Saji. Integrated circuit yield statistics. *Proceedings of the IEEE*, 71(4):453–470, April 83.
- [SBL+86] Charles H. Sauer, Alvin Blum, Paul Loewner, Edward MacNair, and James Kurose. *The Research Queuing Package Version 2*. IBM, 1986.
- [Sch87] Misha Schwartz. *Telecommunication Networks*. Addison Wesley, 1987.

- [Sei84] Charles Seitz. Concurrent vlsi architectures. *IEEE Transactions on Computers*, C-33(12):1247–1265, December 1984.
- [Sei85] Charles L. Seitz. The cosmic cube. *Communications of the ACM*, 28:22–33, 1985.
- [SM82] K. Saraswat and F. Mohammadi. Effect of scaling of interconnections on time delay of vlsi circuits. *IEEE Journal of Solid State Circuits*, SC-17(2):275–280, April 1982.
- [Spe88] Don Speck. Mos vlsi trendlines. Usenet distribution, September, 16 1988.
- [SPP87] W. Richard Smith, Scott Powell, and George Persky. A missing neighbor model for capacitive loading in vlsi interconnect channels. *IEEE Journal of Solid-State Circuits*, SC-22(4):553–557, August 1987.
- [ST83] T. Sakurai and K. Tamaru. Simple formulas for two- and three-dimensional capacitances. *IEEE Transactions on Electron Devices*, ED-30(2):183–185, February 1983.
- [Sut89] Ivan E. Sutherland. Micropipelines. *Communications of the ACM*, 32:720–738, 1989.
- [TH88] Stuart K. Tewksbury and L. A. Hornak. Communication network issues and high-density interconnects in large-scale distributed computing systems. *IEEE Journal on Selected Areas in Communications*, 6(3):587–609, April 1988.
- [Tho80] C. D. Thompson. *A Complexity Theory for VLSI*. PhD thesis, Carnegie-Mellon University, Pittsburg, PA, 1980.
- [UKH85] Kazuhiro Ueda, Hitoshi Kitazawa, and Ikuo Harada. Champ : Chip floor plan for hierarchical vlsi layout design. *IEEE Transactions on Computer Aided Design*, CAD-4(1):12–22, January 1985.
- [WE85] Neil Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design - A Systems Perspective*. Addison-Wesley, October 1985.
- [yF81] Tse yun Feng. A survey of interconnection networks. *IEEE Computer Magazine*, 14(12):12–27, December 1981.
- [YT84] C. P. Yuan and T. N. Trick. Calculation of capacitance in vlsi circuits, 1984.