

Number 786



**UNIVERSITY OF
CAMBRIDGE**

Computer Laboratory

Communication flows in power-efficient Networks-on-Chips

Arnab Banerjee

August 2010

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500
<http://www.cl.cam.ac.uk/>

© 2010 Arnab Banerjee

This technical report is based on a dissertation submitted March 2009 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Girton College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

<http://www.cl.cam.ac.uk/techreports/>

ISSN 1476-2986

Abstract

Networks-on-Chips (NoCs) represent a scalable wiring solution for future chips, with dynamic allocation-based networks able to provide good utilisation of the scarce available resources. This thesis develops power-efficient, dynamic, packet-switched NoCs which can support on-chip communication flows.

Given the severe power constraint already present in VLSI, a power efficient NoC design direction is first developed. To accurately explore the impact of various design parameters on NoC power dissipation, 4 different router designs are synthesised, placed and routed in a *90nm* process. This demonstrates that the power demands are dominated by the data-path and not the control-path, leading to the key finding that, from the energy perspective, it is justifiable to use more computation to optimise communication.

A review of existing research shows the near-ubiquitous nature of stream-like communication flows in future computing systems – making support for flows within NoCs critically important. It is shown that in several situations, current NoCs make highly inefficient use of network resources in the presence of communication flows. To resolve this problem, a scalable mechanism is developed to enable the identification of flows – with a flow defined as all packets going to the same destination. The number of virtual-channels that can be used by a single flow is then limited to the minimum required, ensuring efficient resource utilisation.

The issue of fair resource allocation between flows is next investigated. The locally fair, packet-based allocation strategies of current NoCs are shown not to provide fairness between flows. The mechanism already developed to identify flows by their destination nodes is extended to enable flows to be identified by source-destination address pairs. Finally, a modification to the link scheduling mechanism is proposed to achieve max-min fairness between flows.

Acknowledgements

First and foremost, I would like to thank Simon for giving me the opportunity to work in his lab and for all his support over the last 3 years. This work was funded by a Cambridge University Domestic Research Scholarship.

I am also grateful to past and present members of the Computer Architecture Group for their support. In particular, I would like to thank Bob for his guidance and a myriad of helpful discussions. Bob, Alban and especially Jeong have always listened patiently to my ideas and given helpful feedback. Nick helped with Perl and Andrew gave great advice with implementation issues. Finally, Simon, Andrew and especially Alban deserve extra thanks for taking the time to proof-read this thesis. I am grateful to Pascal Wolkotte and Gerrard Smit at the University of Twente who collaborated with parts of the work described in Chapter 3.

Outside of the Lab, I would first like to thank my parents for giving me the opportunity and support to get a good education and therefore do a PhD in the first place. I would also like to thank my sister and extended family. For comedy, I have always relied on the Dudes. Lastly, but most of all, this PhD would not have been possible without Liza who did everything from cooking my meals to proof-reading my thesis. Without her, I would still be in bed.

Contents

1	Introduction	17
1.1	Future computing architectures	17
1.2	Thesis contribution	17
1.3	Publications	18
1.4	Thesis outline	18
2	Background	21
2.1	Introduction	21
2.2	The design of NoCs	21
2.2.1	Topology and routing	21
2.2.2	Flow control and buffer organisation	22
2.2.3	Buffer management	24
2.2.4	Switch allocator	26
2.2.5	Virtual-channel allocator	27
2.2.6	Router pipeline	27
2.3	Base-case router	28
2.4	Summary	28
3	Power and Energy	31
3.1	Introduction	31
3.2	Related work	31
3.3	Selection of NoC test cases	32
3.4	Router designs	33
3.4.1	Circuit switched router	33

CONTENTS

3.4.2	Wormhole router	34
3.4.3	Quality-of-Service providing virtual channel router	34
3.4.4	Speculative virtual channel router	35
3.5	Power measurement framework	35
3.6	Power measurements	37
3.6.1	Power at fixed throughput	37
3.6.2	Packet energy under no congestion	41
3.6.3	Packet energy under congestion	44
3.7	Performance measurements	46
3.7.1	Uniform random traffic	46
3.7.2	Localised traffic	47
3.7.3	Streaming traffic	47
3.8	Energy-Delay-Product measurements	49
3.9	Area measurements	50
3.10	Summary	51
4	Flows, NoCs and Efficiency	53
4.1	Introduction	53
4.2	On-chip communication flows	53
4.2.1	Historical perspective	54
4.2.2	Flows in future systems	55
4.2.3	Flows in NoCs	56
4.3	Efficiency in NoCs with flows	56
4.3.1	The problem	56
4.3.2	Identifying flows	60
4.3.3	Providing scalable flows support	60
4.3.4	Dynamically allocating flows to VCs	64
4.3.5	Results	68
4.4	Summary	71
5	Fair Allocation to Flows	73
5.1	Introduction	73

5.2	The problem	73
5.3	Background and related work	76
5.4	Approach for NoCs	82
5.5	Identifying source-destination flows	84
5.5.1	Implementation options	85
5.6	Source-count based max-min arbiters	87
5.7	Modifying separable allocators	88
5.7.1	Modifying input arbiters	88
5.7.2	Modifying output arbiters	90
5.8	Results	91
5.9	Future work	95
5.10	Summary	96
6	Conclusions	97
6.1	Thesis summary	97
6.2	Future directions	98
	Bibliography	107

CONTENTS

List of Figures

2.1	A two-dimensional mesh topology with X-Y dimension ordered routing example from node at co-ordinates (0,0) to (1,2).	22
2.2	A high-dimension network topology.	23
2.3	Flow control and buffer organisations, showing message structure on the left, with router organisation and allocation policies on the right.	25
2.4	Principle of operation of a separable allocator. Input arbiters first choose between requests at the inputs with output arbiters selecting a single winner for each output port from amongst these.	27
2.5	A four stage router pipeline.	28
3.1	Router architectures studied.	36
3.2	Link and router power (including data-payload and sideband control) for streaming traffic experiments with the shaded regions showing static power.	38
3.3	Packet energy for streaming traffic.	42
3.4	Packet energy under congestion.	45
3.5	Packet latencies of the dynamic networks for uniform random traffic for varying injection rates.	47
3.6	Packet latencies of the dynamic networks for localised random traffic at varying injection rates.	48
3.7	Packet latencies for combined streaming and uniform random traffic for the GuarVC network.	49
4.1	Flows in Video Object Plane Decoder.	55
4.2	Flows in a HiperLan2 baseband processor.	55

LIST OF FIGURES

4.3	A flow from source S_1 to destination D_1 causes all VCs and buffer spaces to be used up if the transmission rate is greater than the reception rate and flow based packet dependencies are not enforced.	57
4.4	Uniform random traffic latency versus injection rate for base-case router.	58
4.5	Hot-spot traffic latency versus injection rate for base-case router without flow based dependencies enforced.	58
4.6	Translation traffic pattern.	59
4.7	Link utilisation under translation traffic for base-case router without flow based dependencies enforced.	59
4.8	Incorrect dependencies enforced between packets with output queuing when flows do not share a common route.	61
4.9	Regions of individually identifiable through-nodes from the perspective of a central node resulting in varying queueing strategies in a 49 node mesh network.	61
4.10	Dynamic allocation of flows to VCs. Instead of VCs being uniquely associated with individual flows, they are dynamically linked to arriving flows.	63
4.11	Number of destinations for up to 2, 3 and 4 hops away.	63
4.12	Packets of a flow passing through routers in a sequential, pipelined manner ensures that the minimum resource usage policy is enforced.	65
4.13	Percentage of packets sent non-contiguously for 8×8 network with uniform random traffic at varying injection rates.	66
4.14	Optimistic signaling of flow freed event.	67
4.15	VC allocation logic, including VC flow table and mechanisms to set and clear it, at one output port.	67
4.16	Hot-spot traffic latency versus injection rate with flow based dependencies enforced.	69
4.17	Link utilisation under translation traffic with flow based dependencies enforced	70
4.18	Uniform random traffic latency versus injection rate with flow based dependencies enforced.	71
4.19	Link utilisation under translation traffic with flow based dependencies enforced but the number of flows exceeding the number of VCs.	72

5.1	Seven flows passing through a single router, with associated bandwidth demands shown in flits/cycle.	74
5.2	Network setup to evaluate unfair allocations.	74
5.3	Unfair separable allocator does not account for varying number of flows at different input ports contending for the same output port.	76
5.4	Traffic pattern with five sources sending data to a single destination node.	76
5.5	Different traffic utility functions.	78
5.6	An example network setup demonstrating congestion collapse.	79
5.7	Two source-destination flows going to the same destination intersecting at a network node.	85
5.8	Different packet transmission behaviour with flows identified by source-destination pairs and those identified only by destinations, with S_y^x representing packet x from source y.	86
5.9	Basic units of source count selector and adder.	87
5.10	Traffic pattern to highlight potential starvation problem with flows.	90
5.11	Modified input arbiter structure.	91
5.12	Modified output arbiter structure.	92
5.13	Network setup to test fairness mechanisms in input arbiters.	93
5.14	Uniform random traffic latency versus injection rate with SDF router.	95

LIST OF FIGURES

List of Tables

2.1	Base-case router parameters.	29
3.1	Standby power breakdown	40
3.2	Streaming traffic packet energy breakdown.	42
3.3	Energy-latency product for the various traffic types.	50
3.4	Area of the different routers.	51
4.1	Number of VCs required to efficiently schedule varying proportions of traffic going up to different hop counts.	64
4.2	Bit permutation traffic patterns. Each bit d_i of a b -bit destination address is a function of a single bit s_j of the source address, with j being a function of i . . .	70
4.3	Bit permutation traffic results.	70
5.1	Unfair flow allocations in base-case and DF routers.	75
5.2	Unfair flow allocations in base-case and DF router for hot-spot traffic.	77
5.3	Fair allocations in SDF router compared to unfair flow allocations in base-case and DF routers.	93
5.4	Fair allocations in SDF router for hot-spot traffic.	93
5.5	Rate allocations for test to stress input arbiters.	94
5.6	Rate allocations for test with different flow request rates.	94

LIST OF TABLES

Introduction

1.1 Future computing architectures

The vast increases in attainable computation performance seen over the past few decades has been a direct result of the semiconductor industry keeping pace with Moore's Law to provide ever larger numbers of faster transistors. However, the new constraints of power dissipation and communication delays are changing the nature of on-chip computation and communication. In the era of power-limited VLSI, the primary benefit provided by technology scaling is the increased number of transistors at each generation. At the same time, fundamental physical properties mean that while the transistors get faster with scaling, relative to them, long wires become slower. This makes it increasingly difficult to globally synchronise the large number of transistors available. Continued scaling therefore directs us to consider architectures that exploit a high degree of parallelism through the use of many independent and loosely co-ordinated tasks.

Networks-on-Chips (NoCs) are ideal for the communication needs of such tasks. They are a natural evolutionary step towards partitioning and regularity in global interconnect. Traditional ad-hoc global wiring is replaced by an optimised, regular layout of comparatively short wires connected by routing elements.

1.2 Thesis contribution

To date, most NoC designs have been developed and evaluated within the confines of a limited benchmark set of highly bursty, synthetic traffic patterns. Although still important, recent studies of expected future applications have shown that a lot of the parallel communication patterns will be more *streaming* in nature. Whatever communication infrastructure is used, it must therefore be able to provide high performance in the presence of both the more traditional bursty traffic and such stream-like *communication flows*.

Two broad approaches are possible to deal with such flows. The first approach would be for resources to be statically partitioned and reserved for individual flows. The conventional approach taken in NoCs literature has been to consider only such static assignment and treatment of flows. For example, designs have been presented which reserve a particular bandwidth (within circuit-switched networks) or even provide multiple, separate networks for the different traffic streams expected. However, static mapping or over-provisioning of physical network resources is sub-optimal, and at lower traffic levels, leads to reduced system performance. In a given time period when a resource is not being fully used, the notion of exclusive ownership of the resource leads to it being idle – but unusable by other requestors – some proportion of the time.

The alternative option is to provide a single, general-purpose network that can dynamically multiplex all the traffic on to a single set of shared resources. The ability to dynamically re-allocate the shared resources to requesting communication entities removes the above potential

1. INTRODUCTION

inefficiency. Indeed the resulting *statistical multiplexing gain* has been the main reason for the success of many of our current large-scale networks and also the reason behind the continuing push to unify the telephony and data networks.

Much research exists in the field of designing dynamic allocation-based, general-purpose packet-switched NoCs. However, few of these consider the presence of the communication flows now becoming increasingly important. It can be shown that this omission conceals some of the key benefits of a move to a NoC design. The focus of this thesis is therefore to develop dynamic allocation strategies for NoCs that can significantly improve the achievable performance with flow-based traffic. More specifically, the rest of this thesis considers the three primary features of NoC design listed below.

Power The ‘Power Wall’ has become one of the most important constraints in modern VLSI. To ensure that the NoC operates at its most power-efficient point, it is important to first understand the power dissipation characteristics of NoCs. This thesis therefore presents detailed power characterisations of a range of NoC architectures.

Efficiency To maximise attainable performance, a key goal of NoC design has always been to make efficient use of the available physical resources. Complex mechanisms, such as virtual channels and shared buffer architectures have been developed to this end. However, the resource utilisation efficiency can be significantly degraded in the presence of flow-based communication patterns. This thesis develops a scalable mechanism to identify flows and uses this to increase the efficiency of resource utilisation given flow-based traffic.

Fairness In any situation where a resource is shared across a number of users, some concept of a fair division of that resource is essential. In the context of dynamic allocation mechanisms for NoCs, this has resulted in a number of proposals for fair bandwidth allocators. However, most existing designs operate on the basis of individual packets which can be shown to result in unfair allocation to flows. Capitalising on this deficiency, this thesis initiates the development of mechanisms to fairly allocate bandwidth to flows.

1.3 Publications

Parts of the power analysis work of Chapter 3 have been published as [6]. In particular, the referenced paper reports the power analysis for the CS, WH and SpecVC routers introduced in that chapter. An extension of this work, reporting the performance and power-efficiency figures of Chapter 3, is in press as [7] with the characterisations for the GuarVC router being carried out by Pascal Wolkotte and Gerard Smit at the University of Twente. Finally, an exploration of the importance of locality in on-chip traffic has been jointly reported with Daniel Greenfield in [34].

1.4 Thesis outline

Chapter 2 first provides background material into the design of NoCs.

Chapter 3 starts by investigating the power aspects of NoC design. A number of different NoC architectures are accurately modelled to quantify their power and energy needs. The understanding developed about NoC power issues then guides the design approach taken in the rest of the thesis.

Chapter 4 first reviews existing work to show why stream-like communication flows are to be expected in the future. It then shows that a significant source of inefficiency remains due to the absence of flow-aware allocation. A means of identifying flows is then proposed which is used to remove this inefficiency.

The problem of a lack of fair allocation across flows is addressed in Chapter 5. The flow identification mechanisms of Chapter 4 are extended and the router arbitration policies modified to achieve max-min fairness between flows.

Finally, Chapter 6 lists the conclusions and looks at related future work that can be carried out.

1. INTRODUCTION

Background

2.1 Introduction

A fundamental effect of continued technology scaling has been the degradation of the performance of long wires compared to transistors [38]. As feature sizes shrink, transistors become faster, relative to which long wires become slower. Deep sub-micron wires also present further complex challenges of noise and fault-tolerance problems [31]. Such problems lead to the complete un-scalability of large, system-wide buses and ad-hoc wiring.

A common technique to reduce wire delay is to split long wires into shorter segments and provide a driver for each segment. A natural progression from this increases the throughput with pipelining by replacing the drivers with flip-flops. It is then a small step to add some control around each flip-flop to give a network architecture. The irregular, ad-hoc wiring is replaced by a regular placement of short wires interconnected by intelligent routers. At the same time, large monolithic computation units are broken up into smaller tiles. These tiles now communicate with each other by exchanging packets of information over the shared network resulting in a scalable tiled architecture [22]. These represent the fundamental principles behind all *Network-on-Chip* (NoC) designs.

Much research exists in this field with several published textbooks providing a good introduction to it [23; 58]. This chapter provides a similar overview of the fundamentals of NoC design and complements it by presenting some of the latest research findings. More detailed background in the particular fields covered by this thesis of NoC power issues, on-chip communication flows and their impact on efficiency and fairness, are provided as the topics are introduced in Chapters 3, 4 and 5 respectively.

2.2 The design of NoCs

2.2.1 Topology and routing

The particular arrangement of routers and wires used in a network defines that network's topology. Many regular topologies, such as a *2-dimensional mesh*, *torus* or *ring*, as well as many irregular topologies can be used. The 2D-mesh has so far been the most commonly used design, as it naturally maps on to the two-dimensional on-chip environment (Figure 2.1). Recent research has suggested that a higher dimension connectivity (demonstrated in Figure 2.2) such as that provided by a *flattened butterfly* might provide much better performance [33; 48]. However research into these is still ongoing and such high dimension networks are not yet commonly used. Nonetheless, they are expected to become increasingly important in the future.

Once a topology has been selected the next natural question is which path should packets take to traverse the network? Deciding this is the task of the routing algorithm used. All routing algorithms can be classified as either *oblivious* or *adaptive*. Oblivious algorithms

2. BACKGROUND

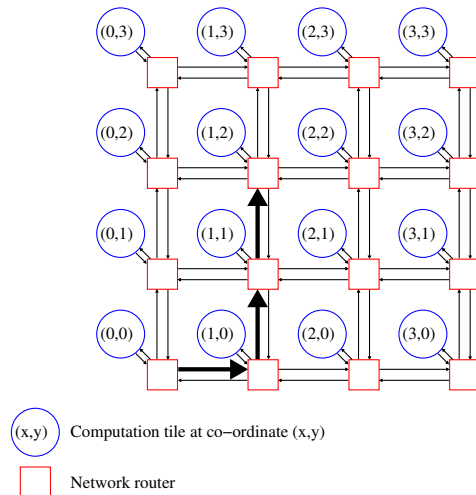


Figure 2.1: A two-dimensional mesh topology with X-Y dimension ordered routing example from node at co-ordinates $(0,0)$ to $(1,2)$.

do not base their route selection on existing traffic conditions in the network. Although this can potentially lead to significant load imbalances across the network (thereby reducing network performance) such schemes have been popular given their simplicity (and hence low implementation costs). For torus and mesh topologies the dimension-ordered oblivious routing algorithm is commonly used. With this scheme a packet completely traverses one dimension first, until its co-ordinate matches that of the destination in that dimension. The procedure is then repeated for all remaining dimensions until the packet reaches its destination (Figure 2.1).

To achieve better load balance adaptive routing algorithms can be used, where the selected route depends on current network traffic or other conditions. Various studies have looked at providing dynamic routing for NoCs [1; 41]. However the concern of increased complexity has limited the take-up of such mechanisms so far.

A variety of mechanisms can be used to implement the chosen routing algorithm. From the network perspective, the simplest option is that of *source-routing*. With this scheme the entire route is computed at the source node and the result appended to the packet. Intermediate network nodes then simply read this information to decide where to send each packet. The alternative is for network nodes to themselves calculate the next hop node of packets on a hop-by-hop basis. Both these mechanisms could be implemented as either a *table-based* mechanism (where a table with an entry for each destination provides the routing result) or an *algorithmic* mechanism (where the route is computed by specialised routing circuits). Especially for oblivious algorithms, algorithmic routing can be achieved at very low-cost.

2.2.2 Flow control and buffer organisation

Efficient resource utilisation has always been one of the key aims of general-purpose NoC design. In the presence of multiple communication requests on-chip, the job of efficiently allocating them to available resources falls to the *flow control* mechanism being employed.

In the constrained on-chip environment, *input-queued* architectures (those with buffering only at the router input ports) are generally preferred as they avoid the need for internal speedup necessary with *output-queued* architectures (those with buffering only at the router

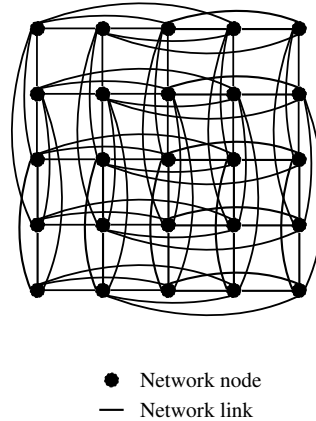


Figure 2.2: A high-dimension network topology.

output ports). The organisation of the input buffers is then intricately tied to the flow control mechanism being used. This section briefly reviews the developments leading up to the current design consensus in these fields.

Circuit-switched flow control: Circuit-switched flow control provides the simplest means of allocating network resources to messages. With this approach a fixed amount of network resources are first reserved from the source to the destination in the *circuit setup* phase. All messages between these end nodes are then routed over these reserved resources. After all messages have been transmitted, the resources are released in the *circuit tear-down* phase.

Packet-based buffered flow control: The difficulty of predicting communications in any environment with dynamically arriving communication requests hinders the performance of circuit-switched networks. Fixed amounts of resources remain reserved for a particular circuit even if the communication demand falls below this level, making poor use of these resources. The initial circuit setup time can moreover add significant delay. Some form of dynamic flow control mechanism is therefore required.

The standard starting point for most dynamic flow control mechanisms is to divide messages into packets and add buffers to the routers to allow these packets to be temporarily stored. This provides much more flexibility in allocating bandwidth as large messages do not have to be dealt with contiguously and in the presence of contention, packets can be delayed as opposed to simply having to be dropped.

Store and forward flow control: When a packet arrives at a router, the simplest option is to store the entire packet in a buffer. Once the entire packet has been received it can be forwarded to its desired output channel. Before being forwarded, the packet must therefore acquire the correct output channel and a large enough buffer in the downstream router. This scheme is demonstrated in Figure 2.3(a).

Wormhole flow control: Wormhole flow control further divides packets into *flits* (short for FLOW control UNITS)¹. The routers' storage buffers are similarly divided into flit sized

¹Flits of a packet can generally be classified as head, body or tail flits. A head flit is the first flit of a packet and usually carries control information. The last flit in a packet is labelled

2. BACKGROUND

units. Buffer space is now allocated on a per-flit rather than on a per-packet basis [21]. This greatly improves buffer usage efficiency as router buffers now do not need to be at least as big as the largest packets; just a few flits worth of buffering is sufficient. In an on-chip environment, where buffer resources are expensive, this is an important gain.

The output channels are, however, still allocated to entire packets. Once the first flit of a packet is granted an output channel it is not released until the last flit of that packet has been successfully forwarded. This scheme is demonstrated in Figure 2.3(b).

Virtual-channel flow control: With wormhole flow control, if a packet blocks before all its flits have been successfully forwarded (perhaps due to a lack of buffer spaces downstream), it will still own the output channel allocated to it. This results in inefficient resource usage, as the same channel could potentially have been used by a different, non-blocked, packet. Virtual-Channel flow control corrects this, by dividing the links into a number of *virtual channels* (VCs). In its simplest form, a fixed number of flit buffer spaces are associated with each VC in every input port in each router. Before making progress, a packet must acquire a downstream VC. Instead of allocating links to whole packets, flits of packets that have successfully acquired VCs then compete for link bandwidth individually [19]. This means that if a packet is blocked, the link can successfully be re-used by flits from other packets and the bandwidth is not wasted. This is demonstrated in Figure 2.3(c).

Shared-buffer organisation: With the fixed partitioning of buffer resources across VCs described above, buffer spaces could either go unused (if more than one packet is not allowed to use a VC at one time) or Head-Of-Line (HoL) blocking could occur (if a packet at the head of a VC queue blocks another packet behind it in the same queue). To overcome these inefficiencies, shared buffer architectures have been developed where buffer spaces are dynamically shared across all VCs [68; 73; 90]. With these schemes a unified pool of free buffer spaces exists at each input port (and potentially across multiple input ports). Only the minimum required number of these are dynamically associated with particular VCs. This is shown in Figure 2.3(d).

Overall, all the mechanisms discussed above can be seen to promote a single allocation policy. They aim to ensure that packets do not reserve any more resources than they need – the *minimum resource usage* policy.

2.2.3 Buffer management

The cost of recovering any dropped packets within the on-chip environment makes packet recovery mechanisms difficult to justify. Instead, the favoured approach is to ensure that packets are never dropped. As part of this, *back-pressure* mechanisms are used to ensure that routers only transmit data if sufficient buffer space is guaranteed to exist in the downstream router. Two of the most commonly used mechanisms for this are described below.

On/Off buffer management: With this mechanism each router holds a single bit of state for every input queue in the downstream router, indicating whether the router is allowed to send data to that queue or not. A single bit signal is sent by the downstream router to flip this state. An *off* signal is sent when the number of free buffer spaces in the input queue goes below a particular threshold and an *on* signal is sent when it goes above

a tail flit and is used to de-allocate any resources held by that packet. All flits between the head and tail are called body flits and are used for transporting the data payload.

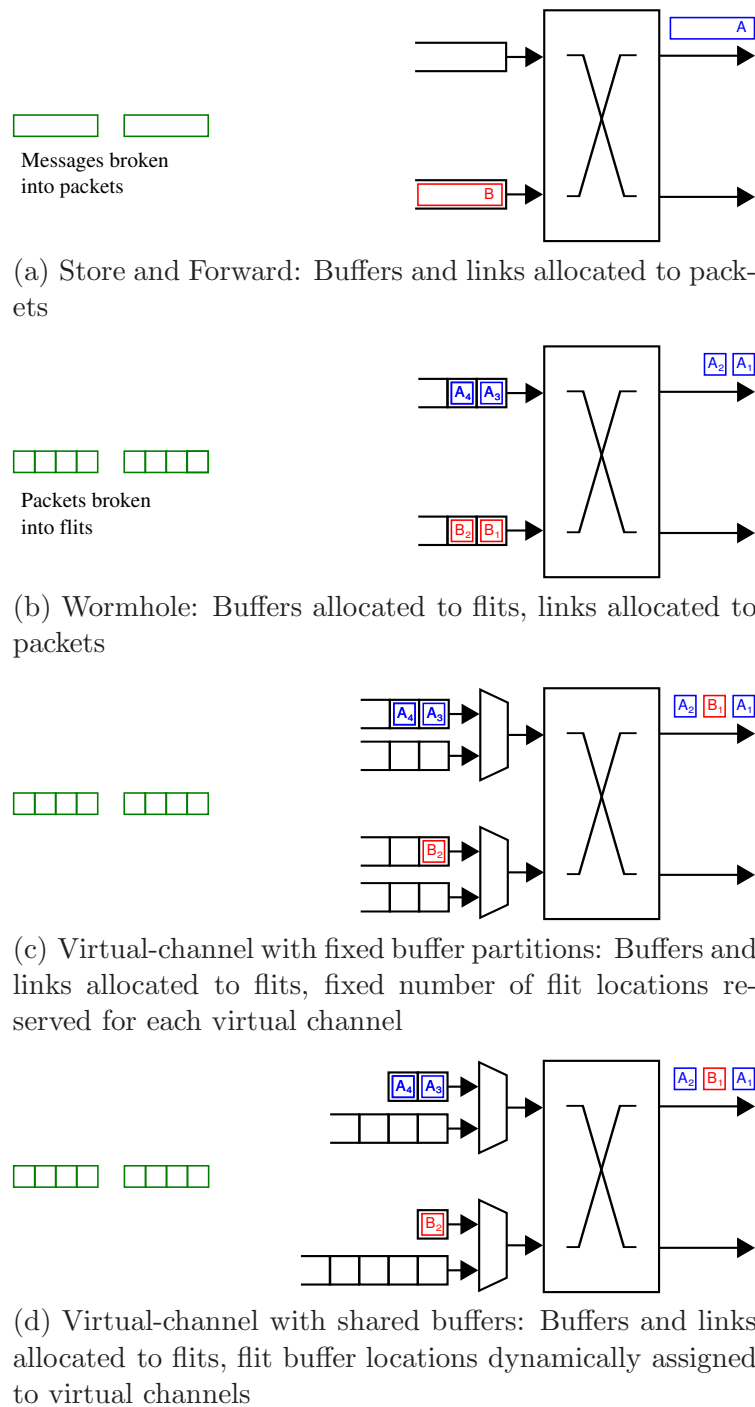


Figure 2.3: Flow control and buffer organisations, showing message structure on the left, with router organisation and allocation policies on the right.

2. BACKGROUND

another. In a simple scheme the threshold levels can be identical. One of the parameters setting the threshold value is the *round-trip-time* (RTT) - the amount of time for the signal to traverse to the upstream node, be processed by it and the pipeline of already sent flits to clear. Careful selection of this threshold ensures that, in the worst case, the upstream router does not transmit more flits than there are free spaces remaining downstream, thereby preventing buffer overflow.

Credit-based buffer management: With this mechanism, every router keeps track of the number of free buffer spaces for each downstream queue. Every time a flit is sent downstream this number is decremented. When the downstream router de-queues a flit from the inputs it sends a *credit* back upstream. On receiving this credit, the upstream router increments its count of free buffer spaces. The upstream router now only transmits flits if its count is non-zero. Credit-based buffer management can be shown to make better use of buffers than on/off flow control but this comes at the cost of increased upstream signalling and state held.

2.2.4 Switch allocator

In any dynamic allocation based network, a mechanism is required in each router to resolve contention in output resource requests from different input packets or flits. This task of allocating access to the crossbar switch and (thereby the output links) falls to the *switch allocator*.

Two of the primary aims of the design of this allocator are to ensure a fair division of the shared resources across the different requesters and maximise the number of satisfied requests. A better achievement of these is mainly traded-off against a lower cost design (with cost measured in terms of delay, power and area).

Allocators are commonly composed of groups of *arbiters*. A single arbiter can arbitrate for a single resource by selecting a single winner from a group of requests for that resource. Much research exists into the design of good quality, low-cost on-chip arbiters with *round-robin* or *matrix* arbiters being commonly used [23]. Round-robin arbiters iterate over input requests in a pre-defined order, skipping over any inputs without an active request. At each step, the first actively requesting input found is selected as the winner. Matrix arbiters select winning requests with a Least-Recently-Used policy. Both of these maintain a prioritised queue of the inputs in a form which requires little computation to convert input requests to output grants for low numbers of inputs.

A good cost-performance balance for allocating across larger numbers of inputs and outputs is achieved by separable allocators. These are formed from individual arbiters arranged into a two stage structure. As shown in Figure 2.4, a set of arbiters for each input port form the first stage of allocation, which select a single winner at each input port. The input-stage winners' requests are then forwarded to the relevant output arbiters arranged into the second stage. These select a single winner for each output port from the input stage winning requests.

It has been shown that lower average transmission delay can be achieved if flits of packets are not interleaved on the output links [50]. Therefore, unless a packet blocks, flits of packets should be sent contiguously, i.e. in a wormhole flow control manner. Part of the simple mechanisms enabling this involves only updating the state of the arbiters when a tail flit is granted.

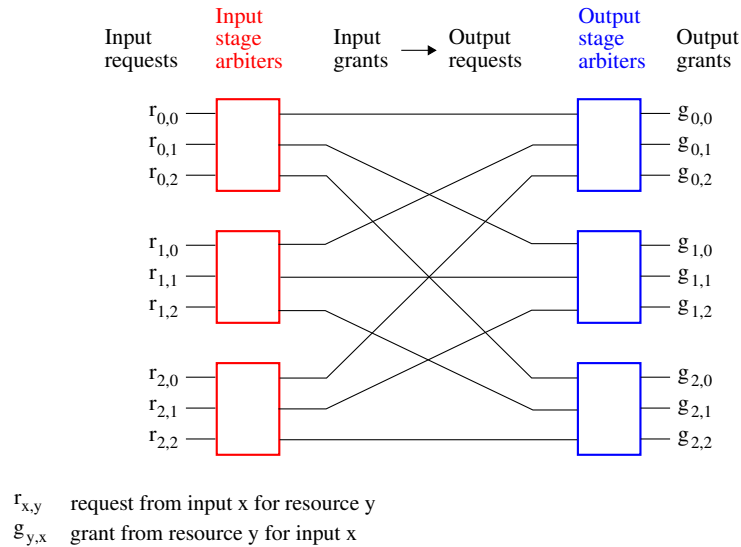


Figure 2.4: Principle of operation of a separable allocator. Input arbiters first choose between requests at the inputs with output arbiters selecting a single winner for each output port from amongst these.

2.2.5 Virtual-channel allocator

In virtual-channel based routers packets need to acquire a virtual-channel before they can be sent to the next router. Traditionally this has been achieved by providing a separable allocator, very similar to the switch allocator structure described above, to allocate output VCs to input packets. Every head flit entering a router generates a request for this allocator. When a successful grant is received, the granted VC is reserved for that packet. The packet can then request for access to the switch to make progress. When the tail flit of that packet is transmitted, its VC is marked as empty and can then be re-used by other packets.

More recently, a much simpler VC allocation mechanism has been developed [50]. Instead of a separable allocator, a single *free-VC FIFO* is provided at each output port. This holds a list of free VCs at that output port. Packets now only generate requests for the switch allocator. Head flits that win access to this output port then simply take the VC ID at the head of this queue. When the same packet's tail flit is transmitted, it re-queues its VC ID at the back of this queue.

2.2.6 Router pipeline

The simplest router pipeline consists of 4 stages as shown in Figure 2.5. When a head flit arrives at a router, a route computation (RC) first indicates which output port it must go to. It then acquires a VC at that port in the VC allocation (VA) stage. Once a VC has been acquired switch allocation follows. A grant from this enables the packet to progress through the crossbar switch in the subsequent switch traversal (ST) stage. If the packet is unsuccessful in acquiring the required resources in the VA or SA stages, it simply retries in the next clock cycle instead of making progress to the next stage.

The importance of low latency on-chip communications have motivated various optimisations to reduce the pipeline depth. First employed by the SGI routing chip was the idea of doing the route computation for one router in the previous hop router - called *look-ahead*

2. BACKGROUND

route computation	VC allocation	switch allocation	switch traversal
----------------------	------------------	----------------------	---------------------

Figure 2.5: A four stage router pipeline.

routing [32]. The result of this computation is carried by the packet to the next hop router. When a packet arrives at a router, the desired output port is therefore already known. This removes the route computation stage from the critical path, thus reducing the pipeline depth to three.

Peh *et al.* have shown how it is advantageous to speculate that a packet will always succeed in acquiring a VC [75]. Given this assumption, the SA stage can be performed speculatively, in parallel with the VA stage, thereby reducing the pipeline depth to two. Additional logic is used at the end of the clock-cycle to check whether the speculation was correct. If not, the speculative switch allocation result is discarded and all the computation repeated. Mullins *et al.* have further shown how both the VA and SA stages can be performed speculatively one clock cycle in advance by assuming that a request will arrive in the next cycle [65; 66]. In the best-case, therefore, a newly arriving packet will find resources pre-allocated for it. It can therefore proceed straight to the ST stage, thereby traversing the router in a single clock cycle. With the FIFO based VC allocator discussed in Section 2.2.5, the separate VC allocator becomes unnecessary and only the switch allocation needs to occur speculatively. The importance of low delay networks has also resulted in alternative single-cycle implementations. Park *et al.* [72] and Kumar *et al.* [50] have demonstrated how the ST in one router can be overlapped with SA in the next router, again resulting in single cycle router delay.

Most NoC router components have been optimised for low delay and it has been shown that even quite complex designs can achieve a clock period between a high performance 20 FO4¹ to modest 30 FO4 delays [66; 75].

2.3 Base-case router

A high performance router, incorporating the most widely accepted parameter choices and the most recent developments in the field, has been selected as the base-case design for the rest of this thesis. In particular, the new designs developed in Chapters 4 and 5 are compared to this base-case. The particular parameter values for this router, with associated reasons for their selection, are specified in Table 2.1.

2.4 Summary

The degradation in performance of wires relative to transistors with continued scaling means that traditional ad-hoc wiring or large, system-wide busses no longer scale to larger systems. Instead, a Network-on-Chip architecture with intelligent routers routing packets over a set of short, optimised wires represents a scalable communication infrastructure.

¹One FO4 (fan-out-of-4) delay is the delay of a single inverter driving four identical inverters. Expressed as multiples of this, the delay of many circuit blocks remains similar between different process generations. Reporting delay in FO4 units therefore represents a process-independent delay metric [97]

Parameter	Value	Reason for selection
Topology	Mesh	Best maps to 2D on-chip environment and most commonly used in existing literature.
Routing	Static, dimension-ordered X-Y	Simplicity and lack of consensus in the research community about dynamic routing mechanisms.
Flow control	Virtual-channel	Proven clear performance benefits.
Buffer organisation	Shared buffer with 8 VCs and 16 flit spaces per input port	Good buffer utilisation, with 16 flit spaces representing current consensus in NoCs community about useful buffer size.
Buffer management	Credit based	Good buffer utilisation.
Switch allocator	Separable allocator with wormhole switching	Most widely accepted architecture with wormhole switching minimising latency.
VC allocator	Free VC FIFO based	Low overhead design.
Pipeline depth	Single cycle	Clear performance benefits with several demonstrated ways of achieving it.

Table 2.1: Base-case router parameters.

2. BACKGROUND

Such a reorganisation of on-chip wiring necessitates many new design decisions with the network topology, the buffer-organisation and the allocation mechanisms representing some of these new parameters. Much research has been done across the large resulting design-space and a vast number of NoC designs are now available to be used. For a general-purpose NoC, a dynamic allocation and packet-based, virtual-channel router with deterministic routing and low delay allocation presents a current consensus design and has therefore been selected as a base-case for this thesis. However, many unanswered questions still remain. The rest of this thesis in particular looks at which of these large number of proposed designs best fit in to the severely power-constrained era and how they need to be modified to support the expected communication patterns of the future.

Power and Energy

3.1 Introduction

For more than 30 years the size, supply and transistor threshold voltage scaling in VLSI has been the key provider of increased computation performance at ever lower power levels. However, a side-effect of this scaling has been an increase in transistor leakage power. This has now grown to the extent that the threshold voltage can no longer be scaled as before. This in turn directly limits supply voltage scaling and hence importantly limits dynamic power reduction. Over the same period, designers have also pushed the use of power-hungry techniques, such as very high frequencies and deep pipelines to maximise attainable performance. This has led to the current state in high-performance chips of very high on-chip power and power densities, further increases in neither of which can now be supported. Power has therefore become a principal design constraint. The continued demand for greater computation performance must be met by increases in power efficiency.

The effects of scaling also mean that interconnect power can take up a growing proportion of the system power budget. Techniques to further increase communication performance, such as inserting buffers, pipelining long wires or moving to NoCs further increase communication power demands. Given this fundamental shift, an accurate understanding of NoC power characteristics is therefore critical. Although it is known that buses and ad-hoc wires simply do not scale and so NoCs must be used, the question of which NoCs to use to obtain maximum power efficiency is currently unanswered. This chapter targets precisely this question by comparing the power, energy and energy-efficiencies of several different NoCs and aims to give a direction to future NoC design work¹.

3.2 Related work

A good overview of the origins and resulting issues relating to the power consumption problems in VLSI is provided by Harris *et al.* [97], Horowitz *et al.* [40] and Mudge [61]. This section provides a summary of existing power characterisation work specifically for NoCs, highlighting the varying approaches taken by researchers so far.

Peh *et al.* have aimed at providing valuable NoC power estimates early in the design cycle by developing a set of high-level power models for a set of NoC router components –

¹This work was carried out in collaboration with Pascal Wolkotte and Gerard Smit at the University of Twente. In particular the power and performance measurements on the GuarVC router were performed by Pascal Wolkotte. The link characterisation work was carried out by Robert Mullins at the University of Cambridge. The parts of this work referring to NoC power and area for the CS, WH and SpecVC routers have been published as [6] and the results for the GuarVC router, along with the performance and energy-efficiency parts are in press as [7].

the resulting framework being named Orion [16; 96]. They went on to use these models to estimate the power of various wormhole and virtual-channel router architectures. Although useful in getting quick estimates, the limited accuracy of the high-level results and the limited number of components currently modelled restrict the usefulness of these data.

Due to the increasing applicability of NoCs at future technology nodes, the changes in NoC power under technology scaling is an important consideration. To this end, a similar power measurement methodology to Orion is provided by Xi *et al.* [101]. High-level power models were again derived and then embedded into a Transaction Level Modelling (TLM) based simulation framework. The authors then used the Berkeley Predictive Technology Model [42] to extend the work to examine the impact of technology scaling. This allowed a key prediction of increasing interconnect power compared to router power to be made for future technologies. However the questionable accuracy of the high-level models still remains a problem.

Banerjee *et al.* [8] aimed to increase the accuracy of power results by first extracting a SPICE level netlist from a synthesised design. This was used to develop power models for NoC router components which provided power results under random traffic simulations. However, only a limited range of architectures were used, limiting the results that could be obtained.

Mullins [62] improved on the accuracy front by fully synthesising and performing place-and-route on a specific router design before utilising the extracted parasitics to analyse its power consumption. This work provided highly accurate power figures and was thus clearly able to demonstrate the importance of low-level ideas such as clock-gating. However, the single architecture used currently stands as its main limitation. A comparative study is clearly necessary to be able to judge the quality of the design and the impact of different design choices on the final power figures.

A further development in terms of accuracy was achieved by Lee *et al.* [53] who fabricate a System-on-Chip with an on-board NoC for communications between varying intellectual-property (IP) blocks. Out of the total chip power of 160mW, 51mW was reported to be consumed by the NoC.

The importance of comparing NoC architectures to the alternative of a bus-based design prompted Dielissen *et al.* [27] to compare Philips' *Æthereal* NoC to a bus-based system. The comparison is performed analytically and is done so for only wires as the router power is assumed to be negligible. Although this allows the authors to provide an important bound on power savings achieved by using NoCs instead of buses, all the other work referenced above shows that router power will certainly not be negligible.

All the work highlighted here shows how much of the existing research has either aimed at high-level (and therefore potentially inaccurate) characterisations of various NoCs or detailed characterisation of very specific NoCs. A detailed characterisation and comparison of a variety of representative NoC designs is clearly lacking. This work aims to provide precisely such a study.

3.3 Selection of NoC test cases

Any comparative characterisation study such as this would ideally evaluate the impact of all input parameters on any output results. However, the large resulting expansion in the design space would make the study completely intractable. Instead it is necessary to select a single or small number of values for many parameters that return a few designs that are representative of a large family of designs. All the parameter values used in the rest of this work were therefore selected using this approach – wherever possible, a single representative value was selected for every design parameter.

Utilising this approach, only fully synchronous routers have been selected for this study. Several asynchronous on-chip network routers have been demonstrated [5; 12]. These designs reduce power consumption primarily by removing clock-tree power but by doing so also provide a different performance level. They represent designs operating at a different power-performance trade-off point. However, many purely asynchronous design approaches can present high delay and area costs [3]. It has been further shown that their inability to sample resource requests synchronously can hurt allocation quality [63]. Instead of using purely asynchronous networks, a consensus design approach is the use of synchronous routers interconnected with asynchronous links to provide benefits of both asynchronous and synchronous design styles [3; 64; 69]. Only minor modifications to existing synchronous router designs are then required, demanding power analysis for purely synchronous routers.

The large number of flow-control methodologies and associated network architectures clearly cannot be represented by a single value. The four networks listed below were instead selected to represent a spectrum of designs, ranging from those that use fully static scheduling to those exploiting the latest techniques to try to achieve the best resource allocation dynamically.

1. The circuit-switched network (from now on referred to as the CS network) presented by Wolkotte *et al.* [100] which has a simple, statically scheduled data-path and no inherent control. This design represents the set of networks that advocate simplicity and static allocation of resources against highly dynamic techniques and place a high importance on meeting *Quality-of-Service* (QoS) demands.
2. A wormhole flow-control and switching based router (referred to from now on as the WH network) which performs dynamic allocation, but not at the cost of highly complex allocation methods.
3. The virtual-channel flow-control based router architecture presented by Kavaldjiev [46] (referred to from now on as the GuarVC network), to allow a comparison of a design using increasing amounts of dynamic control. The router is designed to offer QoS for streaming applications, while also using source routing and semi-dynamic allocation of resources, thus allowing the impact of both of these techniques to be evaluated.
4. The speculative, single cycle, virtual-channel design presented by Mullins *et al.* (referred to from now on as the SpecVC network) [65; 66]. Each router in this design contains a large amount of allocation logic, which attempts to provide good resource sharing, while minimising latencies.

3.4 Router designs

This section provides a more detailed description of the network architectures selected above. For all power measurements, the networks were based on a 4×4 mesh topology with 5-input \times 5-output port routers, with 4 ports connecting to the neighbouring routers and the fifth one connecting to a local computation tile. All flits provided a 64-bit data payload size, with additional control bits necessary for the WH, GuarVC and SpecVC designs. The dynamic networks also utilised a static, dimension-ordered, X-Y routing scheme.

3.4.1 Circuit switched router

The CS router (shown in Figure 3.1(a)) was designed at and obtained from Pascal Wolkotte at the University of Twente. This router provides a simple data-path, being composed only

3. POWER AND ENERGY

of a crossbar with registered outputs. The channel width is 64-bits wide as no control data is necessary. To provide more flexibility, each 64-bit output channel is split into four, 16-bit wide, *lanes*. Given the 5-port design, 20 input and output lanes therefore exist. A 16×20 crossbar provides full connectivity between every input and output lane except that no U-turns are allowed. The crossbar allocation is performed by a configurable memory of 20 entries (i.e. 1 for each output lane), with 5-bits per entry (4 address bits to identify an input lane and 1 valid bit).

The splitting of a 64-bit flit into 16-bit units for transport over the network means that a serialising and de-serialising unit is necessary at the computation tile port of the router (indicated by the *tile-interface* block in Figure 3.1(a)). The completely static nature of the CS network means that a separate control network is necessary to provide all circuit set-up and tear-down functions. To model a scalable solution for this, a simple wormhole routed network was provided. All experiments then considered both the circuit-switched and packet-switched routers, to account for the necessary overhead of the packet-switched network. Further details of this design have been reported by Wolkotte *et al.* [100].

3.4.2 Wormhole router

As no suitable existing design could be found I designed and implemented the WH router (shown in Figure 3.1(b)). This router uses a conventional input-queued architecture with 4-flit deep buffers at each input. A two-stage pipeline is provided. The use of look-ahead routing allows switch allocation to occur in the first stage with crossbar and link traversal in the second.

Control information is appended to each flit rather than being carried in a separate head flit. The 64-bit data-path is therefore combined with a one-hot encoded, 5-bit next-port identifier for look-ahead routing, two bits each for destination X and Y addresses and one bit to identify tail flits, to result in a total flit size of 74-bits.

A pipeline register is provided between the input FIFOs and the crossbar. For the crossbar traversal stage the flit at the head of the FIFO is loaded into this register, which drives it across the rest of the data-path.

Stop-go flow control is used for buffer management, where a *buffer nearly full* signal is output by each input FIFO to the corresponding upstream router to indicate that flit transmission should be stopped.

3.4.3 Quality-of-Service providing virtual channel router

The GuarVC router (shown in Figure 3.1(c)) was again obtained from the University of Twente, having been designed there by Nikolay Kavaldjiev. This router is based on wormhole routing with virtual channel flow control. A conventional input-queued architecture with 4 VCs per port and 4-flit deep buffers for each VC were used. The design provides 2 service classes of guaranteed-throughput (GT) and best-effort (BE).

A 2-bit identifier is used to indicate the VC of each flit. The use of separate head, body and tail flits means that the flit type also needs to be encoded with an additional 2 bits. Combining this with the 64-bit data-path results in a total flit size of 68-bits.

This design uses source routing, where the packet's entire route is determined by the source node and this information then carried by one or more head flits. Six bits are required for each hop of the route – 2-bits for the next port, 2-bits for the VC and a 2-bit identifier for VC allocation. For a 64-bit data-path, routing information for 10 hops are merged into a single head flit.

The crossbar provides an input port for each of the input VC queues, and an output port for each router output port. It is therefore asymmetric with 20 inputs and 5 outputs. This removes the need for a 2-stage separable allocator and creates a single point of arbitration at the output ports which is used to enable QoS needs. To provide for guaranteed-throughput traffic, a central controller allocates network VCs to at most one QoS requiring data-stream. The round-robin arbiters used at each output port then give a predictable arbitration result, where each data stream is guaranteed a certain proportion of the network throughput, i.e. throughput based QoS demands can be met.

Best-effort flows are dealt with by assigning the same VC to multiple data-streams. The additional 2-bit identifier in each header flit is then used to enable fair allocation. A ring counter is maintained in each router and its count incremented by 1 at every cycle. A BE packet is only allowed to make progress in the cycle when its 2-bit identifier equals the counter value. The unpredictable value of this counter at the times when BE packets arrive means that a form of fairness is achieved between different BE packets. However, since BE packets share the same VC, no bandwidth or latency guarantees are provided.

A stop-go flow control method is again utilised to prevent buffer overflow. Further details of this design have been described by Kavaldjiev *et al.* [46].

3.4.4 Speculative virtual channel router

The SpecVC router (shown in Figure 3.1(d)) provides for single cycle flit forwarding by utilising look-ahead routing and speculative VC and crossbar allocation. A conventional input-queued architecture with 4 VCs per port and 4-flit deep cyclic buffers for each VC are used.

Each flit identifies its VC by using a one-hot encoded 4-bit VC identifier. A 5-bit next-port identifier, 4-bits each for destination X and Y addresses and a bit to identify tail flits combines with the 64-bit data-path to result in a total flit size of 82-bits.

Both the VC and switch allocators (based on matrix arbiters) can allocate VCs and crossbar ports speculatively for the next clock cycle if necessary. Since both crossbar and link traversal are performed in a single clock cycle, in the best case, an incoming flit finds pre-allocated resources and can thus be forwarded to the next hop in a single clock cycle.

A stop-go flow control method is again utilised to prevent buffer overflow. Further details have been provided by Mullins *et al.* [65; 66].

3.5 Power measurement framework

The first step in the power measurement methodology was to describe the full network designs in a Hardware Description Language (HDL). A CMOS 90nm, high-performance process with a core voltage of 1.2V and nominal threshold voltage was selected and a standard ASIC tool flow utilised to synthesise, place and route one instance of each of the four routers in this technology. Due to the significant benefits of clock-gating shown by Mullins [62], automatic clock-gating was enabled during synthesis so that low-level clock-gating cells were automatically inserted whenever appropriate enabling conditions were detected. Parasitic extraction was then performed and the results back-annotated into the designs to allow accurate power measurements on the routers.

The inter-router link characterisation was performed separately from that of the routers. The delay and energy figures measured for the links included the link driver, inter-wire capacitance and any repeaters used. Links of length 1.5mm, based on intermediate metal layers (M3-M6), were used. The Quickcap [54] field-solver tool from Magma was then used to extract link capacitance values with an 8-wire model. These were combined with SPICE transistor

3. POWER AND ENERGY

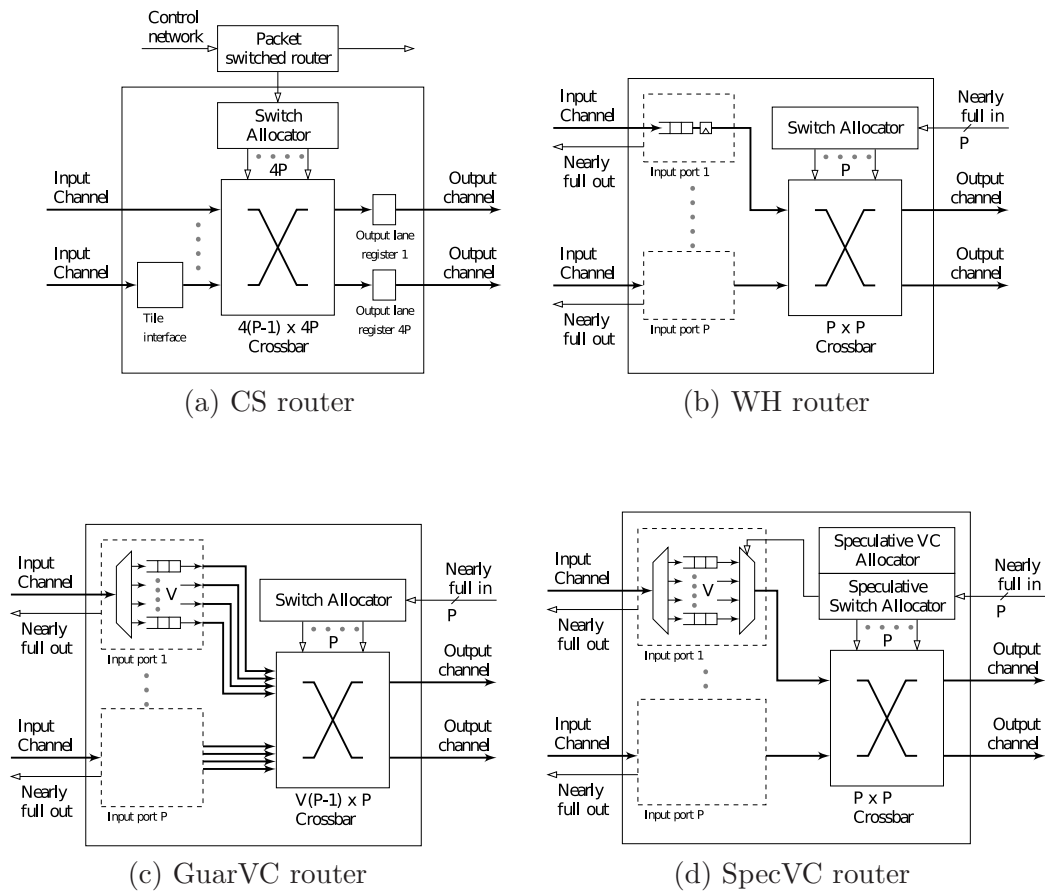


Figure 3.1: Router architectures studied.

models of the wire drivers and repeaters to enable detailed SPICE simulations to be carried out to extract the energy cost of transmitting data on these wires. The energy/delay tradeoffs of various link repeater configurations were analysed. Ultimately, instead of using a delay-optimal repeater configuration, a lower energy configuration with 9.7 FO4 delay with an associated 0.36pJ/transition/mm for the links was selected for this study.

The traffic generation and power and performance measurements of the GuarVC router was carried out by Pascal Wolkotte at the University of Twente. The power measurements followed the same approach as for the rest of the designs, whereas the performance measurements were carried out in an FPGA based simulator [98]. For the rest of the designs the traffic sources were defined entirely in C and the Verilog Programming Language Interface (PLI) utilised to link the C traffic sources with the HDL network descriptions. This provided a highly flexible framework, where each tile could be modelled by a separate set of C routines, at any desired level of complexity.

All simulations were performed at 200 MHz at the nominal Process, Voltage and Temperature corner (PVT).

3.6 Power measurements

3.6.1 Power at fixed throughput

An initial power characterisation was obtained by streaming data through a single router from each of the designs. Four fixed traffic streams were defined, one originating at each of the North, South, East and West ports of the router, with each transmitting a stream of packets to the opposite router port. For each traffic pattern, the switching activity on all internal router nodes was captured and combined with the annotated parasitic information (as described in Section 3.5) to give the *router power*. For every router output port the switching activity on every link was captured and combined with the energy required to transmit a bit on an inter-router link (including the driver and repeaters), again from Section 3.5, to give the *link power*.

256-data-bit packets (i.e. 4 flits in length), each carrying a random payload (resulting in a 50% switching activity factor), were selected for these experiments. The use of the 50% switching activity factor was justified by an observation of this in various applications (such as baseband processing in wireless standards) at the University of Twente. The data rate of each stream was set to a moderate 30% of the maximum bandwidth of a single router link, with flits being sent at randomised intervals. Power analysis was performed for 5000 clock cycles for each experiment. For the CS net, any experimentation requires more detailed consideration as the resource allocation has to be performed separately to utilising the data-path. For simplicity, the resources needed by each stream were only configured once at the start of the experiments. This clearly represents a best-case scenario for this network and must be taken into account when analysing the results. To enable a fair comparison of transporting equal amounts of raw data, the data rate of 30% for the GuarVC router represents the net data rate of only the data payload carrying flits of the packet. An additional three hop header is added to every packet to route it through the router, resulting in a gross data rate of 37.5%. Such streaming traffic patterns are expected to provide useful insight into the expected communications power of a real system for different traffic rates. The results for a single router, presented here, can be used to estimate the power requirements of the entire communications infrastructure. They can also be important from a thermal perspective as power densities determine local heating effects.

Figure 3.2 shows the total, router and link power results of these experiments for all

3. POWER AND ENERGY

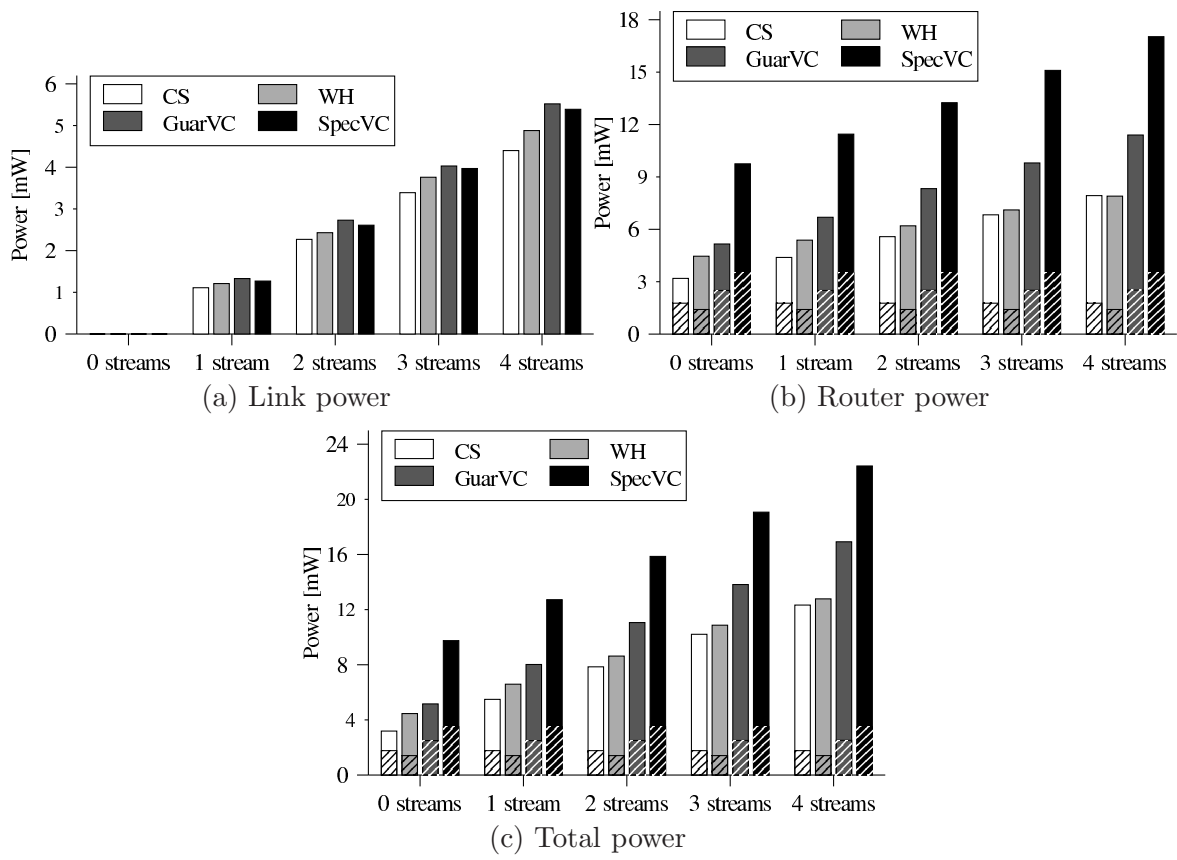


Figure 3.2: Link and router power (including data-payload and sideband control) for streaming traffic experiments with the shaded regions showing static power.

the routers with the leakage power indicated by the shaded areas of the bars. The router powers are comparable to several contemporary computation tiles. For example, a speed optimised ARM926EJ-S processor with caches at 200 MHz consumes 47mW and takes up an area of 1.40mm² [4]. Additionally, the router power is greater than the link power for all the designs and is significantly so for the SpecVC design. The link power results are otherwise as expected with the highest loaded GuarVC links (the extra load caused by the additional head flit per packet) dissipating the most power and the narrowest width CS flits dissipating the least. From these results it is clear that the benefits provided by NoCs come at a high energy cost, at least at the 90nm technology node. Previous work has shown that this high network power cost implies that a 16-node network is indeed not best served with a mesh network and a traditional bus is likely to be lower power [37; 53]. For example, Lee [53] reports around 40% lower energy for a bus compared to a mesh for 16 nodes. With such small networks, the savings in link power are offset by the increases in router power. However, Lee [53], Dielissen [27] and Wolkotte [99] amongst others also show that with larger numbers of nodes, a network architecture quickly becomes the lowest power solution. Lee, for example, reports the crossover point between mesh and bus power to be around 32 nodes [53]. With larger numbers of nodes, although the NoC still demands high power (maintaining the large communication power compared to computation power observation), traditional bus-based or point-to-point links will make even larger power demands.

The increasing number of computation nodes provided by technology scaling, the high power cost of network-based communications and the inability of non-network-based interconnect to provide for global communications importantly point to a new era of communication to computation usage on chip. A reversal of computation and communication utilisation can be expected – whereas in the past, increased communication could be justifiably used to reduce computation, it might now be much more desirable to increase the amount of computation to minimise communication.

All the routers dissipated a significant amount of power even in the 0-stream (i.e. no traffic) condition (from now on referred to as *standby power*). A breakdown of this is reported in Table 3.1. Given the lack of any leakage minimisation techniques, one component of this is leakage power. With the potential for further leakage power increases in future technologies, this component may grow with scaling. Beyond leakage however, there is also some clock related dynamic power. This is caused purely by the activity in the clock tree (since it is only gated at a low-level) and on the clock pins of any non-clock-gated synchronous elements.

Table 3.1 shows that the major contributors to standby power originate along the data-path rather than the control-path. For instance, in the dynamic allocation based routers, not only do the input FIFOs themselves consume a large amount of power, but a large proportion of the clock-tree (another large standby power consumer) goes towards clocking these FIFOs. The impact of this control versus data-path power division is discussed in more detail in Section 3.6.2.

The large observed standby power means that in any real system, standby power reduction techniques will be key. Advanced techniques such as power-gating or the use of high-k dielectrics could clearly be applied to reduce leakage power. Techniques to reduce the dynamic component of the standby power have also been demonstrated, such as the gating of the entire clock-tree demonstrated by Mullins [62]. However, this does not mean that the standby power is irrelevant. This is because all such standby power reduction techniques will be inapplicable in their entirety when packets are being routed. For example, buffers clearly cannot be completely power-gated off while flits are actively stored in them or their clock-tree clearly cannot be deactivated when data is being written or read from them. The importance of standby power is not only that it is the ‘0-traffic’ power but that while routing traffic it

3. POWER AND ENERGY

(a) Dynamic power breakdown

Component	Dynamic Standby Power [mW] [%]			
	CS	WH	GuarVC	SpecVC
Top level clock tree	0.64 , 45.4	1.65 , 53.9	1.52 , 56.9	3.61 , 54.7
Flit buffers + logic	0 , 0	1.27 , 41.5	1.05 , 39.3	1.04 , 15.8
Crossbar	0 , 0	0.09 , 2.96	0.03 , 1.15	0 , 0
Crossbar allocator	0.11 , 7.80	0.05 , 1.64	0.01 , 0.39	0.67 , 10.2
VC allocator	-	-	-	1.05 , 15.9
Output ports	-	-	-	0.20 , 3.07
BE router	0.22 , 15.6	-	-	-
Tile interface	0.44 , 31.2	-	-	-
Other	0 , 0	0 , 0	0.06 , 2.26	0.02 , 0.33
Total	1.41 , 100	3.06 , 100	2.67 , 100	6.60 , 100

(b) Leakage power breakdown

Component	Static Standby Power [mW] [%]			
	CS	WH	GuarVC	SpecVC
Top level clock tree	0.02 , 1.13	0.04 , 2.84	0.03 , 1.20	0.11 , 3.49
Flit buffers + logic	0.10 , 5.65	0.93 , 66.0	1.86 , 74.4	2.30 , 73.0
Crossbar	1.00 , 56.8	0.28 , 20.0	0.36 , 14.4	0.01 , 0.33
Crossbar allocator	0.06 , 3.39	0.09 , 6.38	0.21 , 8.4	0.18 , 5.71
VC allocator	-	-	-	0.23 , 7.30
Output ports	-	-	-	0.20 , 6.35
BE router	0.15 , 8.47	-	-	-
Tile interface	0.33 , 18.9	-	-	-
Other	0.10 , 5.66	0.06 , 4.26	0.04 , 1.6	0.12 , 3.82
Total	1.77 , 100	1.41 , 100	2.50 , 100	3.15 , 100

Table 3.1: Standby power breakdown

forms the fixed base beyond which any routing activity increases the power demands. The large value of this standby power relative to the active power can therefore strongly impact the power-efficiency of NoCs. It is therefore important to strive towards architectures with inherently low standby power needs.

3.6.2 Packet energy under no congestion

A more fundamental metric than the power at a given throughput is the energy required to perform a certain amount of communication in each of the four NoCs. As discussed in Section 3.6.1, the activity of routing a packet causes the router to dissipate additional power specific to the computation performed for each packet, on top of the fixed standby power. Considering the standby power to be the overhead power of a particular architecture means that the *increase* in energy demands caused by routing one packet represents the *dynamic energy cost* of the particular computation performed for each packet. Measuring the increase in a particular router’s power under the 4-stream traffic condition beyond the 0-stream power, multiplying by the simulated time (5000 clock cycles) and dividing by the number of packets processed in that time by that router allowed this dynamic energy cost per packet to be calculated. This method will only work given effective low-level clock-gating and the results obtained showed this to be generally true. With this method, some inter-packet dependencies will inevitably exist (for example, if two packets affect the same clock-gating enable signal for any register), but these are reduced in Section 3.6.3 by utilising more random traffic.

Figure 3.3 shows the dynamic energy cost required to route a 256-bit (i.e. 4-flits), random payload packet through each of the four routers, with a breakdown across the major components reported in Table 3.2. An anomalous clock-gating event was observed at the input ports of the WH router. This meant that the 45.72pJ value calculated with the above method is not directly representative of the buffer computational energy and it was therefore determined to be 66.0pJ with separate experiments (which is then consistent with the other router buffer energy results). Due to the two classes of traffic supported by the GuarVC router (best-efforts and guaranteed-throughput), this router was tested with both traffic types. The lack of a requirement of a unique head flit per packet for the GT traffic means that the GT packet energy is 22pJ lower for the router and 20pJ lower for the link compared to BE packets. No new information is provided by considering a detailed breakdown of the energy for both types of traffic and Table 3.2 therefore only provides the breakdown for the BE traffic.

The key results to note here are that the total energy usage of the different designs are not vastly different and that the data-path components dominate over the control elements in all designs, especially in the CS and WH networks. Specifically, the flit buffers consume a large proportion of the total energy. Moreover, it is interesting to see that the buffer energy is not directly proportional to the amount of buffering in the designs. This is because, with effective clock-gating, the energy needs are only proportional to the computation activity. In the case of the buffers, energy is only required when data is written into or read out of a buffer position. The remainder of the time, clock-gating ensures that very little energy is dissipated. The same explanation also holds true for the rest of the router’s computation activity.

Besides using the dynamic energy cost to compare architectures with the same functionality, it can also be used to compare functionalities across different network types. For example, in the particular case of the buffers, the WH router buffers flits twice, taking 66.0pJ, which is consistent with the single buffering operation performed by the CS router taking 28.1pJ. As the SpecVC router also buffers flits just once, the extra energy for the SpecVC buffers comes from the wider buffers and the more complex data-path around the buffers providing added functionality (as each flit needs to fan out to each register of each VC, unlike the other non-VC

3. POWER AND ENERGY

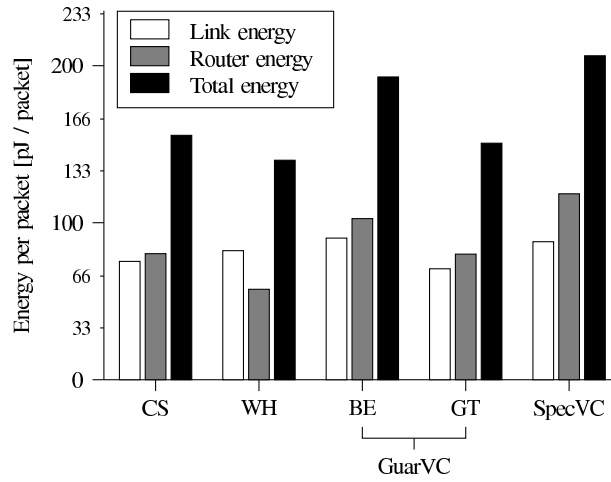


Figure 3.3: Packet energy for streaming traffic.

Component	Dynamic Packet Energy [pJ] [%]			
	CS	WH	GuarVC	SpecVC
Top level clock tree	0 , 0	0 , 0	0.01 , 0	5.98 , 5.10
Flit buffers + logic	28.1 , 47.7	45.72 , 79.4	64.10 , 62.5	58.2 , 49.2
Crossbar	37.9 , 35.4	6.62 , 11.5	31.69 , 30.9	15.5 , 13.1
Crossbar allocator	4.65 , 5.90	2.94 , 5.10	2.84 , 2.80	7.27 , 6.10
VC allocator	-	-	-	1.92 , 1.60
Output ports	-	-	-	8.99 , 7.60
BE router	0 , 0	-	-	-
Tile interface	0 , 0	-	-	-
Other	8.77 , 11.0	2.33 , 4.00	3.98 , 3.90	20.5 , 17.30
Total	79.4 , 100	57.61 , 100	102.61 , 100	118.4 , 100

Table 3.2: Streaming traffic packet energy breakdown.

designs). The GuarVC router also buffers flits just once and has a smaller buffer width, but extra energy is consumed by the extra header flit per packet and added functionality around the buffers similar to the SpecVC design. Similarly, the reason for the high CS energy is clearly the higher order crossbar used for that design. The CS crossbar takes significantly more energy than even the more complex input port multiplexer and crossbar structure used by the SpecVC design.

At this stage it is especially interesting to note the similarity in both the power and energy values of the CS and WH routers. The dynamic switch allocator of the WH router does not make huge energy demands and the extra buffering energy demands by it are clearly offset by the more complex crossbar in the CS router. In a choice between these two designs therefore, no power arguments can be made against the WH design.

The leakage power variation was seen to be insignificant across all the streaming traffic conditions, i.e. the leakage power is practically independent of routing activity. This means that an equivalent *leakage energy cost* for a packet does not exist. Even if some leakage reduction techniques were used, it would still be more meaningful to consider them as standby power reduction techniques and the leakage power as part of the fixed standby power. However, as already discussed, Table 3.1(b) shows the leakage power to also be dominated by the data-path components.

Importantly, from an energy perspective, the much higher data-path power compared to control-path power justifies the use of dynamic allocation techniques for NoCs. As long as the data-path can be kept simple, NoC routers with complex allocation techniques can feasibly be deployed without significantly straining the power budget. The small increase in power that comes from the more complex control can be tolerated, given the better performance and utilisation of communication resources they provide. The high communication to computation ratio discussed in Section 3.6.1 therefore also applies to within the network itself. These arguments are reinforced when considered in the context of reducing transistor cost, given continued scaling. However, given the non-power optimised designs considered here, it is difficult to accurately judge how the data-path to control-path energy ratios might change. On one hand, several data-path optimisations such as the use of SRAMs as FIFOs can clearly reduce the data-path energy. On the other hand, it is questionable whether roughly the order of magnitude data-path to control-path energy difference observed in this work can be eliminated, especially in the context of even larger data-path widths expected for future technologies. Other design aims might also prohibit the use of a very simple (and hence low power) data-path. For example the GuarVC router is forced to use a more complex crossbar to satisfy QoS demands.

The comparative study of the different designs here can also be used to estimate the impact of changing some of the design parameters. For example, the use of three different crossbar architectures here shows the large impact of using a higher order topology on a single router's power needs. However, it is primarily foreseen that the data presented here could be used to better calibrate existing analytical tools, such as Orion [96], which can themselves be used to predict the effect of parameter changes.

The power and energy values reported so far appear to suggest that the WH design is the ideal network choice. However, the most suitable network can only be selected after comparing the performance achieved by the different networks. This is reported in Section 3.7 and extended in Section 3.8 to report the power-efficiency of the designs and hence allow the selection of the best network. However, even without performance considerations, the inability of the WH design to cater for different, independent classes of traffic means that it may be power-inefficient. Many applications identify different classes of traffic and demand different service levels for them [13]. More critically, servicing multiple classes of traffic independently

is an important technique used to avoid message-dependent deadlock [36]. The only way of achieving support for multiple, independent traffic classes with the WH net would be to replicate the entire network. This represents a static partitioning of the resources. The resulting inefficient resource utilisation can result in lower overall power efficiency (as discussed in Chapter 1). Virtual-channel networks on the other hand can associate different classes of traffic with different VCs and dynamically schedule these on to a single set of shared physical resources. This can result in more efficient resource utilisation. The result is that, although the VC routers considered in this study individually demand more power than the WH router, using them may still result in lower overall system power.

3.6.3 Packet energy under congestion

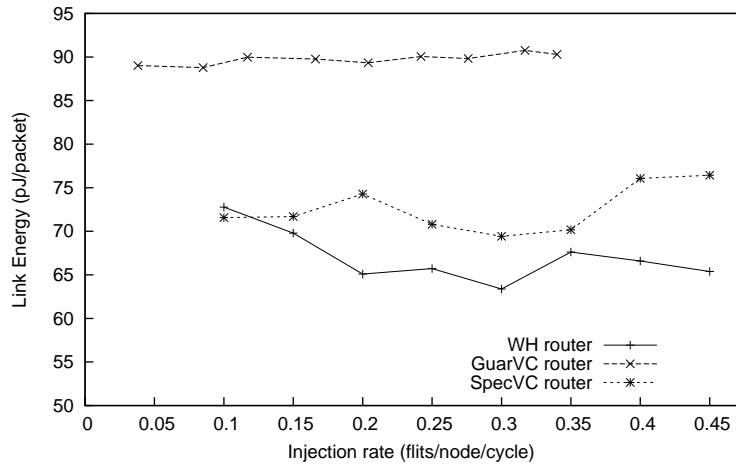
The packet dynamic energy cost reported in Section 3.6.2 does not account for any network congestion. Clearly it is of interest to see how this parameter will affect packet energies. For the WH, GuarVC and SpecVC routers, this was achieved by instantiating 4×4 mesh networks for each design. A traffic source connected to each router then injected random traffic at varying injection rates into the network, destined for random destinations (excluding itself). The inter-packet interval was determined by a Bernoulli distribution. Packets, with each carrying a 256-bit (i.e. 4-flits) random data payload, were again used. An initial 500 clock cycles were used as *warm-up time*, with the next 5000 clock cycles forming the *sampling time*, any packets transmitted during which were the only ones considered in the analysis. A further 300 clock cycles of *drain time* were used to allow any packets transmitted near the end of the sampling time to reach their destinations. A single router, at co-ordinates $x = 2, y = 2$ was considered and the energy of any packets going through it was calculated in the same fashion as in Section 3.6.2.

For the CS network, the current lack of dynamic circuit set-up and tear-down support means that this form of congestion energy experiment cannot yet be performed.

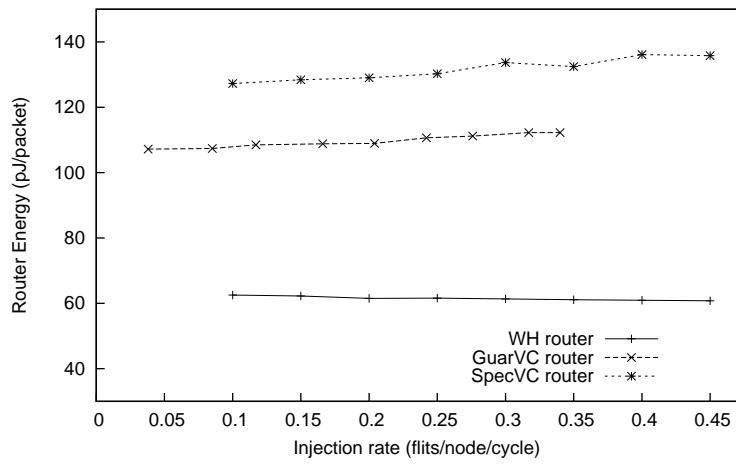
Figure 3.4 shows the packet energies for various injection rates for the three dynamic routers. The link energies were seen to vary much more than the router energies across different experimental runs. This is because the link energy is determined not just by the contents of a flit, but also the contents of the previous flit or any idle periods before the flit. Since in these experiments these are also random variables, the overall variance is further increased.

For the routers the energy per packet was seen to vary very little as network traffic increased. As already discussed, this calculated value represents the energy required to perform the computation specific to the forwarding of one packet through the router. The above result is then intuitively meaningful as effective clock-gating ensures that the amount of data-path computation (the main energy consumer) does not change with congestion. The data-path functions are independent of the amount of time packets spend in network queues. Indeed, a breakdown of packet energies across all data-path components confirmed that their energy demands do not significantly change. The main reason for the SpecVC and GuarVC router energy increase was seen to be due to an increase in allocation and flow control activity, given the increased resource contention. This is a novel result, showing that although performance (such as packet latency) might be seriously degraded at high congestion, there is no considerable direct impact on packet dynamic energy (given effective clock-gating).

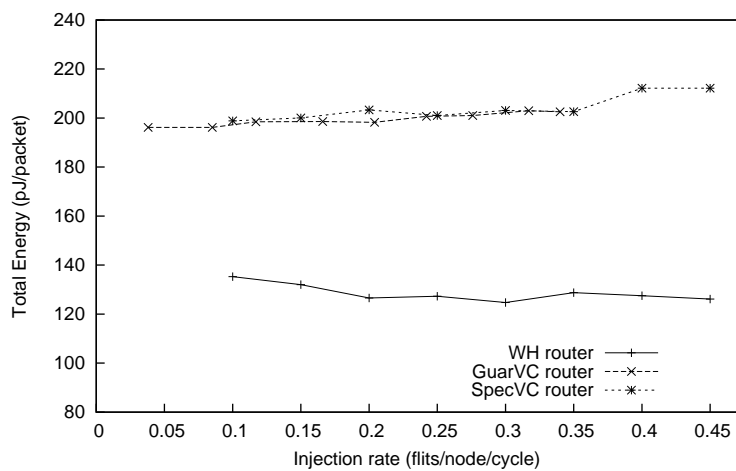
The more important energy impact would come from reducing the time available to effect standby power minimisation techniques. For instance, leakage power reduction techniques such as power-gating of the input FIFOs cannot be fully applied while active flits are stored in the routers' buffers (another reason why standby power represents a key parameter).



(a) Link energy



(b) Router energy



(c) Total energy

Figure 3.4: Packet energy under congestion.

3. POWER AND ENERGY

The above work has shown that network power demands can be quite accurately divided into two fixed quantities – a constant standby power and a relatively constant dynamic energy cost per packet. Once these values are known, good NoC power estimates can be obtained from much simpler functional simulations. Rather than running expensive full micro-architectural simulations, simply obtaining time-stamps of packet forwardings from functional simulations is sufficient to allow the power to be well estimated.

3.7 Performance measurements

The power metrics reported so far only represent one aspect of the designs. For a more complete characterisation it is also important to investigate the performance metrics of the designs. Ideally, such measurements would be obtained using full system-level simulations with real applications. However, no such full system simulators are currently available and synthetic traffic generators have therefore been used. As with the selection of network parameters in Section 3.3, the particular performance characterisation tests performed attempt to represent a range of expected realistic scenarios.

The first test therefore used uniform random traffic. This can be considered to represent any sufficiently complex system. Secondly, the importance of locality, highlighted by work such as that by Greenfield *et al.* [34], prompted results to be gathered for a traffic pattern where the destination of randomly generated communications favoured those nodes closer to the source. Finally, the prevalence of streaming traffic patterns in the important future application classes of media, radio and scientific processing prompted the use of streaming traffic patterns with QoS demands.

All experiments were carried out with an 8×8 network, with 4-flit long packets. An initial 500 packets transmitted per node were used to initialise the network in the *warm-up* period. The subsequent 3000 packets were the ones used during the measurements in the *simulation* period, with an additional *drain* period used at the end to allow all simulation period packets to be received. For the CS network, the current lack of dynamic circuit set-up and tear-down support means that this form of randomised traffic performance measurements were again not possible.

3.7.1 Uniform random traffic

Figure 3.5 shows the measured average packet latency at varying net traffic injection rates¹ into the network under a uniform random traffic pattern. Each source has an equal probability of transmitting to any other source (apart from itself). As expected, the VC networks saturate² at a higher injection rate than the WH network and the delay optimised SpecVC network achieves a lower delay than the WH network. The counter based allocation policy of BE packets used by the GuarVC router was seen to cause its increased delay. With this scheme packets can be stopped at input ports, waiting for the allocation counter value to become the same as their ID even if no other packets are contending for their required output port.

¹As described previously, the net injection rate is set by the injection rate of only the data payload carrying flits.

²A network is said to saturate when the injected load exceeds its traffic carrying capacity. A sharp increase in latency is then observed as the traffic injection queues at the entry points to the network begin to grow without limit.

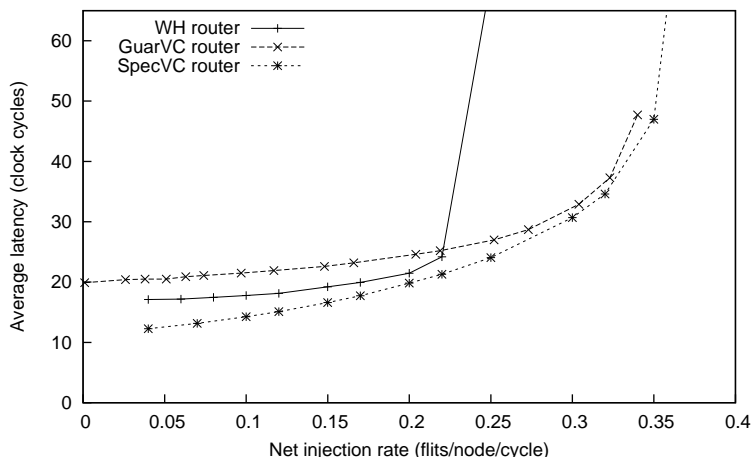


Figure 3.5: Packet latencies of the dynamic networks for uniform random traffic for varying injection rates.

3.7.2 Localised traffic

It has been shown that locality can be an important part of on-chip communications. To model this, a roughly exponentially distributed hop count based traffic generator was used at each node. Based on the empirically observed distribution by Greenfield *et al.* [34], 40% of all transmitted packets were sent only one hop away, 25% were sent two hops away, 15% were sent three hops away and the rest uniformly distributed across the rest of the network. Figure 3.6 shows the measured average packet latencies at varying net injection rates into the network for this localised random traffic.

Compared to the uniform random case, all the architectures now show a lower latency and higher saturation point due to the lower average hop count. In the absence of contention, the packet delay in any NoC router can be broken down into a *head flit delay* and a *serialisation delay*. The head flit delay represents the in-router delay seen by any head flit as it progresses through the router pipeline. Once a head flit has been transmitted, the rest of the flits in the packet will simply follow in the subsequent clock cycles. The time required to send these flits (in these experiments 1 cycle per flit) is called the serialisation delay. The total network delay is therefore represented by equation 3.1. For the WH and GuarVC routers the head flit delay dominates over the serialisation delay. Therefore, compared to the uniform random distribution, these two routers benefit most from localised traffic, where the serialisation delay is lower.

$$T_{total} = H_{average} \times t_{head} + \frac{L}{b} \quad (3.1)$$

where T_{total} is the total network latency, t_{head} is the head flit delay for a single router, L is the message length and b is the link bandwidth.

3.7.3 Streaming traffic

Both of the previous two tests used random traffic patterns with only a single, best-effort, class of traffic. Many radio, multimedia and scientific applications contain much lower dimension and static communication graphs, potentially with QoS demands. Testing with such streaming

3. POWER AND ENERGY

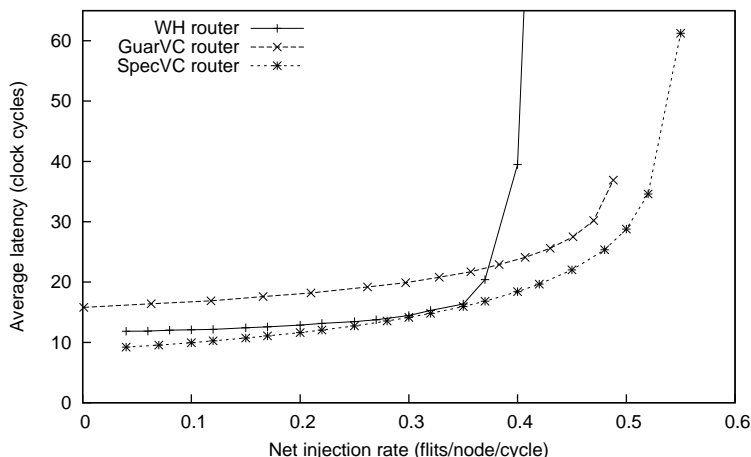


Figure 3.6: Packet latencies of the dynamic networks for localised random traffic at varying injection rates.

traffic is therefore also important. Since the GuarVC design is the only one that provides support for both best-effort and guaranteed throughput classes of traffic, it was the only design tested in this way.

In this experiment, 50% of all traffic from a node was defined to be of type GT and the rest BE. The BE packets followed the same uniform random distribution as in Section 3.7.1 due to the unpredictable nature of the control such packets might represent. The GT traffic from each node was directed towards a single other node. In any real system, the GT traffic is likely to be mapped to the tiles to optimise locality. The distribution of the GT traffic's transmitter-receiver distance was therefore chosen as the same as in Section 3.7.2 – 40% of all pairs were one hop apart, 25% two hops, 15% three hops and the rest four or more hops apart.

Figure 3.7 shows the latency results for this experiment for both BE and GT traffic for the GuarVC network. The uniform random traffic latency for the GuarVC network is also included as a reference. Tests with a different ratio of BE to GT packets resulted in comparable results.

As would be expected, the latency for the GT traffic is significantly lower than for the BE traffic. The pre-allocated VCs for the GT traffic ensure that they are guaranteed a minimum allocated bandwidth. As they also do not need to wait for the counter based mechanism like the BE packets do, they can make progress as soon as the required output port is free, ensuring low delay transmissions. At higher traffic load, increasing contention causes the higher observed latency. The fixed reserved resources for the GT traffic ensure that it doesn't saturate at the same time as the BE traffic. The fewer total number of BE flits injected also means that the BE traffic saturation point is higher than in the uniform random case. As the proportion of GT traffic is decreased this BE saturation point moves closer and closer to the saturation point for uniform random traffic. Conversely it goes further away for increasing proportion of GT traffic.

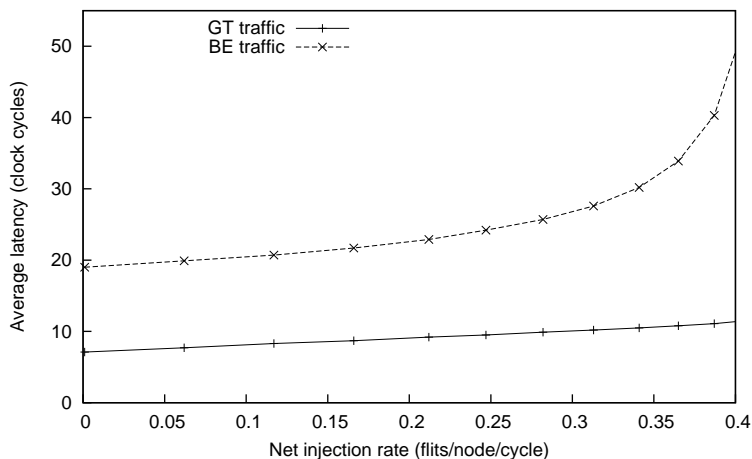


Figure 3.7: Packet latencies for combined streaming and uniform random traffic for the GuarVC network.

3.8 Energy-Delay-Product measurements

One of the key metrics for comparing different designs is the energy-efficiency they provide. The energy and performance modelling of the dynamic networks performed up to now provides sufficient data for their energy efficiencies to be calculated. In NoCs literature the term *delay* is usually reserved to represent the delay of just a head flit through a network. Since the term *latency* is more commonly used to represent the total packet latency the term *Energy-Latency-Product* (ELP) will be used from now on. For each of these networks, their energy efficiency at a single injection rate could be obtained by multiplying the observed latency at that point from the experiments of Section 3.7 by the total dynamic packet energy cost from Figure 3.4(c). The need for both lower energy and latency means that a lower value of this figure represents a more energy efficient design. To convert to a metric where a higher value represents an improved design the figure of $(\text{energy} \times \text{latency})^{-1}$ will be used instead. Clearly though, the ELP figure at a single injection point is of limited value as different applications will make different latency and throughput demands. As such, all the latency values up to the saturation throughput are important. The ELP metric discussed above does not account for this limit up to the saturation throughput. In order to do this, it is proposed that the individual $(\text{energy} \times \text{latency})^{-1}$ values at each injection rate be summed between a fixed lower end of the injection rate and the saturation throughput (r_{sat}). In the limit, this becomes the integral of the curve, or the area under it. Equation 3.2 below shows how this metric is calculated.

$$IELP_{sum} = \int_0^{r_{sat}} \frac{1}{E(r) \cdot L(r)} dr \quad (3.2)$$

where $E(r)$ and $L(r)$ are the packet's energy and latency at the net injection rate r .

Table 3.3 shows this inverse ELP sum ($IELP_{sum}$) metric for the three networks and the three applied traffic types described in Section 3.7. It is important to note that this figure must still be considered in the context of the fixed standby power in the network. Moreover, this metric also does not account for any parameters apart from energy and latency. For instance, if QoS demands were to be quantified the GuarVC router might receive a significantly higher

3. POWER AND ENERGY

Network	$IELP_{sum} [\times 10^7]$			
	Uniform random	Local random	Streaming	
WH	8.9	21.6	-	
GuarVC	- BE	6.8	12.8	8.5
	- GT	-	-	29.7
SpecVC	9.6	19.4	-	

Table 3.3: Energy-latency product for the various traffic types.

score. Finally, the energy value used in this metric is only valid for a single hop. However, the overall $IELP_{sum}$ value for the network can be obtained by multiplying the reported values by the average hop count seen by the packets. Since this average hop count is the same for all the networks (given the same traffic pattern and routing strategy), this product is not reported here.

Looking at the high efficiency of the GuarVC router with GT traffic clearly demonstrates the benefits of specialisation. For the non-specialised, general architectures of the WH and SpecVC routers an important question is whether the additional investment in energy for the SpecVC design brings at least a proportional increase in the performance. The energy-performance ratio provided by the $IELP_{sum}$ figure for the WH and SpecVC routers for the uniform random and local random traffic patterns in Table 3.3 can answer precisely this. The near identical value for both the networks for each traffic pattern implies that the SpecVC design does indeed make a performance return directly proportional to its additional energy investment. Clearly it can be expected that, beyond some point, continued increases in router complexity will not produce proportional performance returns, at which point the efficiency metric will decrease. Similar logic has been extensively applied by Jouppi for microprocessor design to develop the concept of *Microprocessor Efficiency Eras* [44]. In the Chip Multi-Processor (CMP) environment, such observations motivate the use of the most capable microprocessors, which still operate in the highest efficiency region. In the exact same way, the $IELP_{sum}$ metric can justify the use of SpecVC-like designs instead of WH-like designs for on-chip communication networks. Clearly though, the very small number of synthetic traffic patterns and NoC architectures used in this study do not present enough data points to allow the generalisation of this argument to be widely effective in the NoCs field. However, this line of work forms an important future direction in aiding the selection of NoC designs for future systems.

3.9 Area measurements

The area of each router represents another important metric and is therefore reported here. The area of the designs was obtained post place and route and is reported in Table 3.4.

The breakdown of the area for the CS router showed that the high order crossbar is its largest component, being approximately $3.5\times$ larger than the WH crossbar. This, along with the serialisation logic, the packet switched network and the configuration memory areas of the CS router together outweigh the savings of area from reduced buffering compared to the WH

Component	Area [mm ²] [%]			
	CS	WH	GuarVC	SpecVC
Flit buffers + logic	0.008 , <i>7.19</i>	0.045 , <i>57.7</i>	0.160 , <i>74.7</i>	0.166 , <i>67.3</i>
Crossbar	0.058 , <i>53.6</i>	0.016 , <i>20.1</i>	0.034 , <i>16.1</i>	0.015 , <i>6.19</i>
Crossbar allocator	0.004 , <i>3.34</i>	0.005 , <i>6.55</i>	0.018 , <i>8.36</i>	0.015 , <i>6.24</i>
VC allocator	-	-	-	0.022 , <i>8.89</i>
Output ports	-	-	-	0.012 , <i>4.66</i>
BE router	0.009 , <i>8.76</i>	-	-	-
Tile interface	0.026 , <i>24.3</i>	-	-	-
Other	0.003 , <i>2.79</i>	0.012 , <i>15.7</i>	0.002 , <i>0.87</i>	0.016 , <i>6.72</i>
Total	0.108 , <i>100</i>	0.078 , <i>100</i>	0.214 , <i>100</i>	0.246 , <i>100</i>

Table 3.4: Area of the different routers.

router. The increase in area for the virtual channel routers are also caused by the extra input queues for each VC. Furthermore, the GuarVC router has to pay an increased area price with its larger asymmetric crossbar and the SpecVC design requires more area for its speculative allocation mechanisms.

Note that the power, performance and area figures reported so far also allow other parameters such as power-density or area-efficiency of the designs to be calculated. However the focus on power and performance of this study means that these figures are not reported here.

3.10 Summary

This study has presented an accurate power characterisation of a range of NoC architectures by considering a static circuit-switched (CS) network, a wormhole (WH) network, a semi-dynamic virtual channel (GuarVC) network supporting QoS and a speculative virtual channel (SpecVC) network. All designs were synthesised, placed and routed in a CMOS 90nm, high performance technology. Utilising the extracted parasitics then allowed accurate power results to be obtained.

A set of streaming traffic conditions were first used to characterise the power dissipation rates of the routers. The router power was seen to be a significant overhead beyond the link power and also appears comparable to contemporary computation units. These results then significantly point to the existence of a new era of computation versus communication costs on-chip. In some cases, it may now be prudent to perform more computation to optimise global communications.

All the designs dissipated significant standby power produced mainly by leakage and clock-tree power. This standby power can be considered to be the overhead required by a particular architecture and highlights the need to use efficient architectures combined with standby power reduction techniques, to obtain power efficient designs.

The additional power dissipated while routing a packet was used to calculate a dynamic

3. POWER AND ENERGY

energy cost per packet. The much wider data-path compared to the control path meant that it dominated the energy needs, with the buffer energy forming a significant proportion of this figure. Significantly, this result shows that the new computation to communication trade-off extends to within the communications network. They justify the use of complex control in NoC routers.

Calculating the packet dynamic energy cost under congestion for the WH, GuarVC and SpecVC routers showed no significant variation in this value under different traffic levels. This was again seen to be caused by the data-path dominating the energy cost. Since effective clock-gating ensured that the data-path computation did not significantly change with congestion, the energy cost of this did not change either.

For the packet switched routers, performance analysis demonstrated the effects of various trade-offs in the router designs. Dynamic allocation and virtual channels as implemented in the SpecVC design greatly reduced the packet latency under random packet injection while the specialised, QoS supporting design of the GuarVC router offered low latency for GT traffic.

The measured energy and latency results were finally combined into an $IELP_{sum}$ metric that represents the efficiency of a router architecture for a specific traffic scenario. The specialised design of the GuarVC router allowed it to have a high efficiency value for GT traffic. Critically, for the more general WH and SpecVC routers, the efficiency metric showed that the additional power investment made by the SpecVC router resulted in directly proportional performance returns. In the microprocessor domain, the related work of *Microprocessor Efficiency Eras* [44] justifies the use of the highest performance microprocessors that still operate in the highest efficiency regions for CMP systems. Similarly, the router efficiency results favour the use of SpecVC-like designs over WH-like designs to provide the highest performance at no reduction in the power efficiency.

Finally, the area reported for the four routers means that the impact on the full system of using these NoCs can be easily evaluated.

Flows, NoCs and Efficiency

4.1 Introduction

The design of any network demands an understanding of the traffic patterns expected to use that network. For NoCs, likely to be part of future computing architectures, this necessitates gaining an understanding of expected future applications. Until recently, such knowledge has been severely lacking. However, more recent research in this field indicates that stream-like communication flows will be important. This chapter begins with a demonstration of why flows are to be expected. It then moves on to show how current mechanisms for achieving one of the key goals of NoC design – efficient resource utilisation – fail in the presence of flows. A design to enable identification of flows is then developed to solve this, which also acts as an enabling mechanism for the fairness issues discussed in the following chapter.

4.2 On-chip communication flows

Until now, most general purpose NoC designs have been evaluated using (and hence implicitly designed for) a small number of simple, synthetic traffic patterns, with the uniform random benchmark being particularly commonly used. Most of these place a strong emphasis on highly bursty traffic, with no, longer term, static patterns. It is precisely these, more static, communication *flows* – long-lived data transfers between a fixed source-destination pair – that are now becoming important¹. Flow based communications are an inherent property of many applications. Consider a task-graph representation of an application with vertices representing currently active computation kernels and edges representing communication between the various computation kernels (for example as in Figures 4.1 and 4.2). For many applications the graph structure can be highly dynamic and vary across a large number of states over the application’s lifetime. Mapping such an application on to a distributed computation platform results in random and highly bursty communication between different computation tiles. However, for many other applications the representative graph structure can be much more static with a small number of patterns observed most of the time. Mapping these applications on to a hardware platform results instead in repeating communication between fixed source-destination nodes, i.e. communication flows.

The origin of such flows can primarily be attributed to the existence of coarse-grained parallel computation tasks processing large data-sets, arriving in the form of streaming inputs [67; 88]. Some types of such systems have existed for a long time and a thorough survey of older stream processing research is presented by Stephens [88], with the oldest systems traced back to at least as far as the 1960s. A large volume of more recent work has also looked at

¹Note that the traditionally used term *streams*, where traffic is usually considered to be perfectly predictable, can be considered a special case of flows, where some degree of unpredictability still exists.

providing support for stream processing across the board, ranging from language and compiler to architectural support [80; 92].

The following section provides a similar overview of existing flow-like communication patterns and their impact on system design for the family of applications likely to be prevalent in the future. This leads on to the subsequent section which discusses expected future applications and their communication patterns.

4.2.1 Historical perspective

In most older systems, flows originated due to the presence of large volume data transfers. For large memory transfers, perhaps the most well-known mechanism developed is Direct Memory Access (DMA), which sets up a streaming communication pattern between a fixed source-destination pair [74]. The basic need for large memory transfers ensures that DMA continues to be present in much more modern architectures. For example, it is heavily used in the Cell processor to transfer data into the local storage of any of its Synergistic Processing Elements (SPE) [39]. DMA has also featured in NoC design. For instance, Bolotin *et al.* define an explicit traffic class in the design of their QNoC aimed at DMA and other similar traffic types [13].

Another area of traditional computing systems that has seen flows due to large data-transfers has been graphics. Off-chip, the real-time and changing nature of a displayed environment results in the Graphics Processing Unit (GPU) regularly demanding data from main memory. Special hardware support had been provided as soon as necessary with standards such as the Accelerated Graphics Port (AGP) to enable high bandwidth data input into the GPU [83]. Within the GPU itself, the computation has traditionally followed a highly pipelined model. This naturally creates more static communication flows between the different pipeline processing units. Architectures such as Pomegranate which use a shared interconnection network for all on-chip communications therefore see more static communication flows over this network [30]. Currently, advanced graphics is becoming more widespread, as applications continually aim to provide better user interaction and increased support for digital media [26]. Many of the applications designed for this, such as MPEG4 encoding/decoding algorithms, are characterised by continuous data streams [26; 57; 81]. Figure 4.1 (adapted from [87]) gives such an example, showing the flowing communication patterns of a Video Object Plane Decoder (VOPD). Graphics related traffic such as this has been one of the main motivators for the design of application specific NoCs [18; 86; 102].

Recently, there has been a drive towards achieving a ubiquitous presence of mobile network connectivity [9]. In the field of radio, there has been a strong pressure to integrate the signal processing elements on-package or on-chip [24; 79]. Commercial products, such as those by Picochip have also aimed at providing systems for baseband processing [76]. An analysis of the traffic characteristics of radio processing applications has therefore been considered important. Among others, Wolkotte *et al.* [100] analyse the communication patterns of three contemporary radio protocols and conclude that they all exhibit semi-static communication flows. Given the usually continuous nature of transmission/reception events, and the commonly used pipeline based processing approach, this is to be naturally expected. An example of flows in a HiperLan2 baseband processing engine is shown in Figure 4.2 (adapted from [100]). Indeed the same authors and others, such as in [76], go on to build a statically scheduled communications architecture for these systems.

Scientific computing has developed as a parallel field over much of the history of computing. Historically, these applications have been written with explicit communication, standardised by the development of the Message Passing Interface (MPI) [85]. Amongst others, Vetter

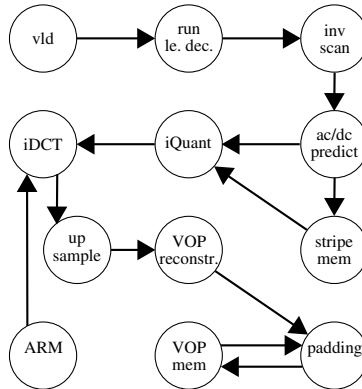


Figure 4.1: Flows in Video Object Plane Decoder.

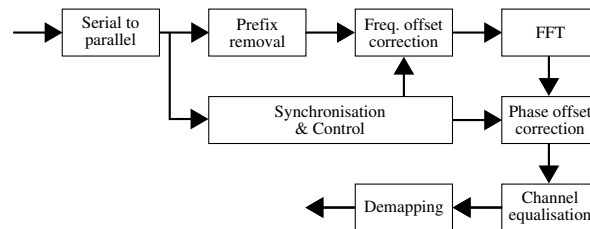


Figure 4.2: Flows in a HiperLan2 baseband processor.

et al. [94] have shown that in many such applications point-to-point communications are responsible for the majority of data transfers. Vetter *et al.* [94], Zamani *et al.* [103] and Kim *et al.* [49] have moreover shown that for many programs, the number of unique destinations for packets from the same source node is quite low. This is the exact opposite of high dimension communication graphs obtained by uniform random synthetic traffic and implies the presence of more static communication flows. Such flows are especially well identifiable when the nodes frequently communicate with each other, as shown by Kim *et al.* [49]. Fundamentally, scientific programs usually work with a representation of a potentially large physical system. Often, the system can be divided into smaller regions with similar, and relatively fixed computation steps needed for each region. This can result in a data parallel and pipelined model of computation [11]. This, more static, computation style can therefore result in similarly static communication patterns.

4.2.2 Flows in future systems

It is widely believed that a new era of computing will soon be with us. Commonly referred to as Tera-Scale Computing, it will be characterised by huge amounts of data [29]. Interpreting this data in terms of high level models will then be a strong focus of future computing applications. Significant research, especially that done at Intel, has supported the idea of *Recognising* models from presented data, *Mining* for that model in the vast stores of data and *Synthesising* models to communicate results back to the user [11; 17; 29]. This forms the RMS suite which is expected to encompass most future applications. The large memory transfers, scientific, media and mobile connectivity classes of applications discussed previously feature

strongly within this suite. As a result, the flow-based traffic patterns fundamentally associated with them are set to become increasingly common. Indeed, where designs have been attempted for future-like highly parallel systems, the flow-based traffic nature has been explicitly utilised. This can be seen for instance, in the circuit-switched network or prediction engines for aiding cache coherency in [28; 43; 55], the architecture designed for *software circuits* developed as MIT's RAW and Tiler's Tile64 microprocessors [91; 93] or the algorithms developed for extracting non-exact solutions from streaming data-sets described in [67].

4.2.3 Flows in NoCs

The aim of any general purpose computation hardware design is to maximise the achieved performance given a limited set of resources. The limited resource constraints include a limited number of transistors, the limited number of wires on a single chip or most critically the limited amount of power that can be supplied to the chip (as discussed in Chapter 3). To maximise attainable system performance, all system components, including NoCs, must maximise the performance obtained with the limited number of resources they are given – they must operate at the most efficient point.

Given that flows are a common feature in emerging applications, as discussed in Section 4.2, application of Amdahl's Law, optimising the common-case, demands that NoCs maximise the resource utilisation efficiency in the presence of flows. The abundance of circuit-switched NoC designs illustrates how most designers until now have aimed at supporting flows by statically allocating fixed amounts of resources to them [43; 76; 91; 100]. However, given that flows are rarely wholly predictable, inefficient resource utilisation results when the bandwidth demanded by a flow falls below its allocated rate. This strongly motivates the design of a NoC that can dynamically multiplex flows onto a shared communications infrastructure. Such dynamic allocation results in *statistical multiplexing gain* – reflecting much better resource utilisation [15]. Moreover, despite the importance of flow-based traffic it certainly cannot be guaranteed to be the only traffic pattern ever seen. Enabling efficient dynamic scheduling of flows within NoCs already capable of efficiently scheduling non-flow-based traffic is the focus of this chapter.

Some limited work has attempted to deal with flows in dynamic allocation based NoCs. Walter *et al.* identify flow-based problems originating at the boundaries of NoCs and propose a solution using only higher-level entities, external to the NoC [95]. With this approach, the lack of NoC operations implicitly at the level of flows adds delays in detecting and responding to flows and therefore represents a non-optimal solution. Dally *et al.* therefore attempt to support flows in a dynamic allocation based network, but they present a design aimed at satisfying worst-case needs [20]. With a large number of nodes this becomes un-scalable and therefore inapplicable for large future systems. A typical-case design approach is therefore key. A similar work to this chapter is presented by Pirvu *et al.*, where they empirically observe the benefits of restricting VC allocation [77]. However, they again do not place a direct emphasis on flows thereby failing to develop a full model of flow-based dynamic networks with an associated inability to design for the typical-case or extend other services to flows.

4.3 Efficiency in NoCs with flows

4.3.1 The problem

Most previous work in NoCs has treated every packet as independent of every other. Subsequent development has then tried to ensure that only the minimum required resources are

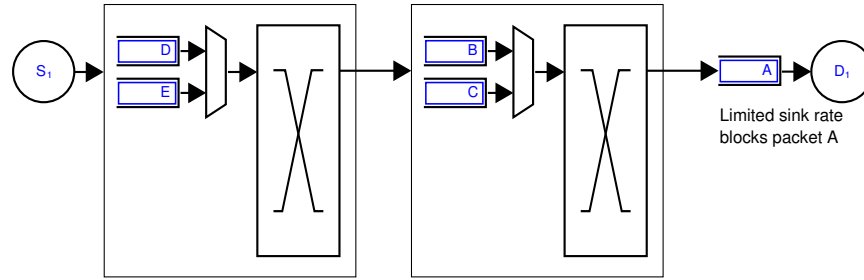


Figure 4.3: A flow from source S_1 to destination D_1 causes all VCs and buffer spaces to be used up if the transmission rate is greater than the reception rate and flow based packet dependencies are not enforced.

reserved per packet at any time, as summarised in Section 2.2.2. This ensures that blocked packets do not keep hold of resources they cannot use. Other packets are free to use these resources to make progress instead – resulting in efficient resource utilisation.

Given the presence of flows however, not every packet can be considered independent. Since, by definition, packets of the same flow are part of the same communications entity, the order in which they are sent sets an implicit dependency between them. If a packet of a flow blocks, no advantage is gained by allowing packets behind it to bypass it – packets of a flow should be received in the order in which they are transmitted. If re-ordering occurs in the network, it will in fact necessitate the use of extra resources at the receiving end to re-order the packets into the correct order.

From a network utilisation perspective, not utilising these packet dependencies undermines the ‘minimum necessary resource usage’ aim discussed previously. With current implementations, when a packet of a flow blocks, the other packets behind it will bypass it to acquire the alternative VCs and their buffer spaces in all the routers along their path, as shown in Figure 4.3. Since the packets are related, whatever caused the first packet to block is also guaranteed to block all subsequent packets. Given this certainty the use of these parallel buffers is unnecessary and is hence beyond the minimum necessary resource usage level. Packets of other flows are then prevented from utilising these resources, driving down overall system efficiency.

Figures 4.4, 4.5 and 4.7 illustrate this problem in the base-case router (described in Section 2.3) with two examples. First, Figure 4.4 shows the latency results with an 8×8 node network, with all nodes injecting 4-flit packets to uniform random destinations at varying injection rates. The measurements reported are for 2000 *measurement packets* sent by each node after an initial transmission of 400 *warm-up packets*. As can be seen, the traffic saturates near an injection rate of 0.35 flits/node/cycle. Figure 4.5 then used the same setup except that 5% of traffic from each node was addressed to a single hot-spot node at co-ordinates (3,3). The predictability of these hot-spot packets means that they represent a long term flow. Figure 4.5 separately shows the average latency of all the packets going to the hot-spot node (labelled hot-spot traffic) and the average latency of all packets not going to the hot-spot node (labelled non-hot-spot traffic) at varying injection rates. As can be seen, the hot-spot traffic saturates at an injection rate of around 0.24 (when the hot-spot destination receives traffic at a rate of around 0.76 flits/cycle). However, since no flow based dependencies are enforced, the hot-spot packets then acquire all the other VCs and buffers on their path, thereby blocking all the non-hot-spot traffic too. This traffic therefore also saturates at the same injection rate, much earlier than the case illustrated in Figure 4.4.

The previous experiment shows a flow based efficiency problem caused by a limited ac-

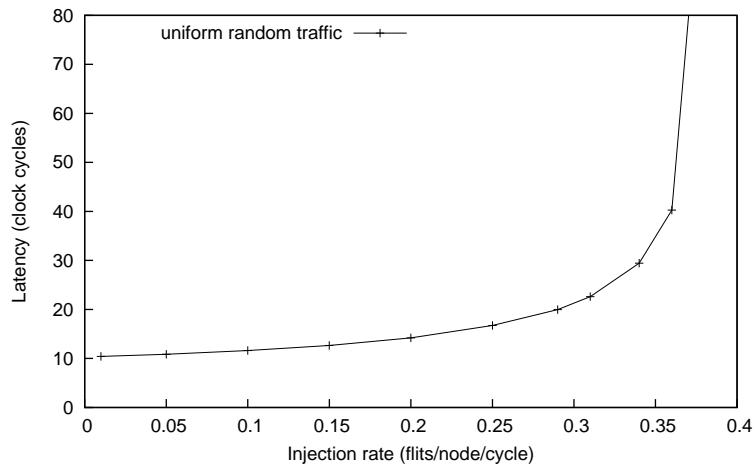


Figure 4.4: Uniform random traffic latency versus injection rate for base-case router.

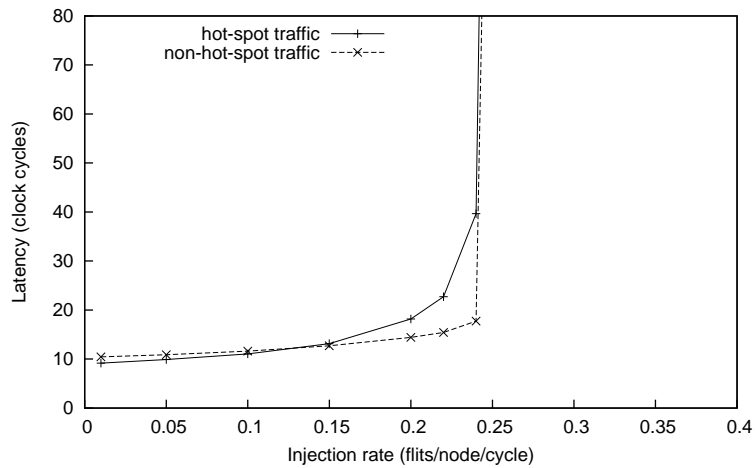


Figure 4.5: Hot-spot traffic latency versus injection rate for base-case router without flow based dependencies enforced.

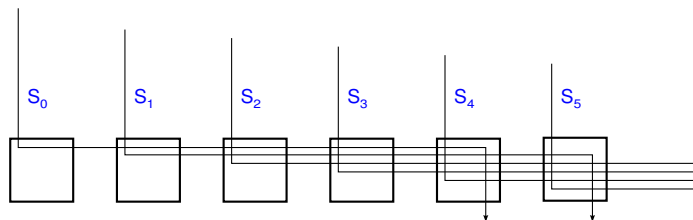


Figure 4.6: Translation traffic pattern.

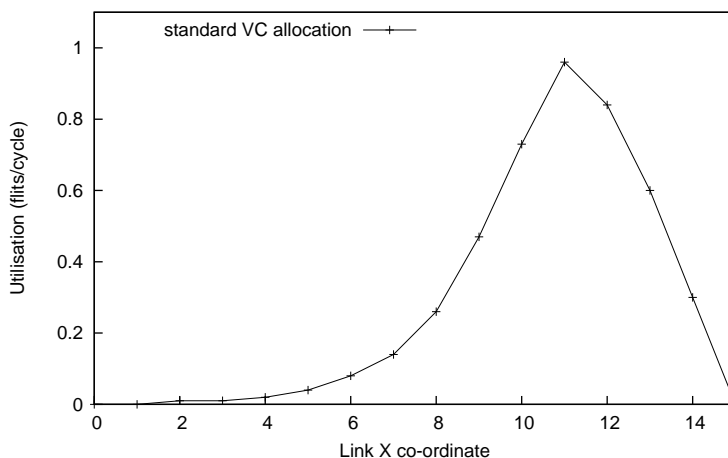


Figure 4.7: Link utilisation under translation traffic for base-case router without flow based dependencies enforced.

ceptance rate at a sink node. Efficiency problems can also be caused by limited flow rates at a network node. Figure 4.6 shows a traffic pattern that highlights this problem. Here, each node sends traffic at a high injection rate to a destination 4 nodes to its right¹. For a 1-dimensional, 16 node network with each node injecting traffic at 0.3 flits/cycle the resulting utilisation of all the *east* links (i.e. how many flits on average are forwarded by that link per cycle) is shown in Figure 4.7. As can be seen, the link utilisation drastically falls for the more westward links. The problem occurs because multiple flows share the same link bandwidth, with the total bandwidth demand exceeding the link capacity. Flows therefore receive less bandwidth than they demand. However, since flow packet dependencies are not enforced, new packets keep being injected into the network. These take up all network VCs and buffers, preventing flows further upstream from making progress.

A possible solution to the above problems might seem to be to use adaptive routing. However, in many cases this will not work as it does not solve the fundamental problem of packets of a flow being injected at a faster rate than at which they can be sunk. It will simply move the problem to a different set of links.

¹This can be considered to be an adaptation of the classical tornado traffic [23] to the mesh topology.

4.3.2 Identifying flows

Before a network can operate with a concept of flows, it must first be able to identify flows. For many applications flows could be identified directly by the programmer or by static analysis by a compiler (for example in the case of the completely static task graph of Figure 4.1). Sufficient information exists at these higher, software levels to enable flows to be identified. A mechanism to enable these high-level entities to label communications with different flow IDs and inform the network of these would clearly make the network flow-aware. This represents a more complex interface between the hardware and software. The software provides explicit information about its communications¹, as opposed to injecting generic ‘traffic’. However, such complex hardware-software interfaces are still in the early development stage. Further development of these is beyond the scope of this work. Instead the framework developed here uses a simpler, more common hardware-software interface. The data injected into the network consists simply of packets destined to particular network node addresses with the network identifying flows based solely on this information. However, the framework developed can be easily extended to work with alternate interfaces and this stands as important future work.

In solving the efficiency problems of Section 4.3.1 with the hardware-software interface discussed above, the first question that arises is what the relationship between different packets should be that defines them to be part of the same flow? Such an ability to identify flows is the first fundamental step before any flow-based allocation methodologies are developed – nothing can operate on units of flows if flows cannot even be identified. The most widely used architecture that attempts to answer this is *output queueing* [45]. The hypothesis here is that, within a router, the only packets that depend on each other are those going to the same output port. Each router therefore provides a queue at each output port. However, from the wider network perspective, this technique only captures the correct packet dependencies in the special case when flows blocking at a node use the same route from there onwards. Figure 4.8 shows an opposing situation, where a blocking flow blocks an alternate flow with which it does not share an entire path. The eventual implication is that flows should in fact be identified by their final destination node – resulting in *end-node queueing* [20]. A single reception point exists at the final destination and if it blocks all packets destined to it are guaranteed to block.

Output queueing can in fact be seen as a limiting case of end-node queueing. Output queueing effectively classifies packets in terms of their next hop address, whereas end-node queueing classifies them according to their final destination address. A clear continuum exists between these two ends – flows could be identified by a node address N-hops away from their current position (Figure 4.9). This is important in the presence of localised traffic, when it is known that the majority of traffic will only go up to N-hops away. This is discussed further in Section 4.3.3. However, given that this represents a limited-case of end-node queueing, the general-purpose framework demonstrated here identifies flows by their final destinations.

4.3.3 Providing scalable flows support

This section begins the development of a hardware mechanism to enable flow relationships between packets to be enforced to make buffer utilisation efficient. Rather than proposing a solution that is tied to a single design, the aim is instead to develop a framework that can be applied to a variety of NoCs. Moreover, as discussed in Chapter 3, the general approach is to only modify the control-path to ensure efficient power utilisation. Identifying flows by destinations fits well with this approach as any routing mechanism will require packets to

¹An example of this is the use of different *message-classes* to avoid deadlock.

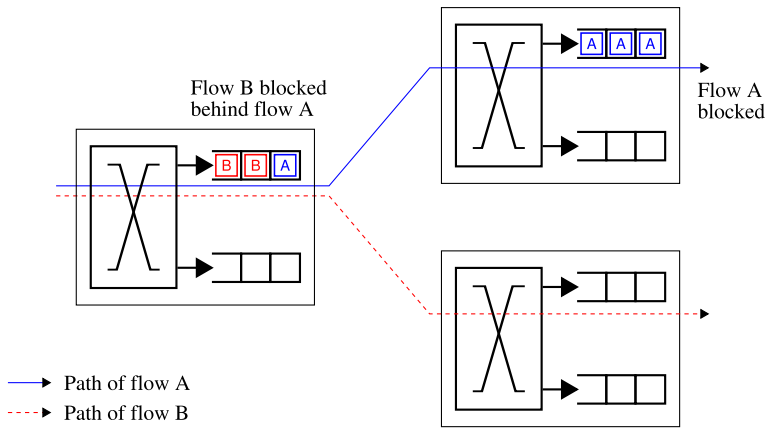


Figure 4.8: Incorrect dependencies enforced between packets with output queuing when flows do not share a common route.

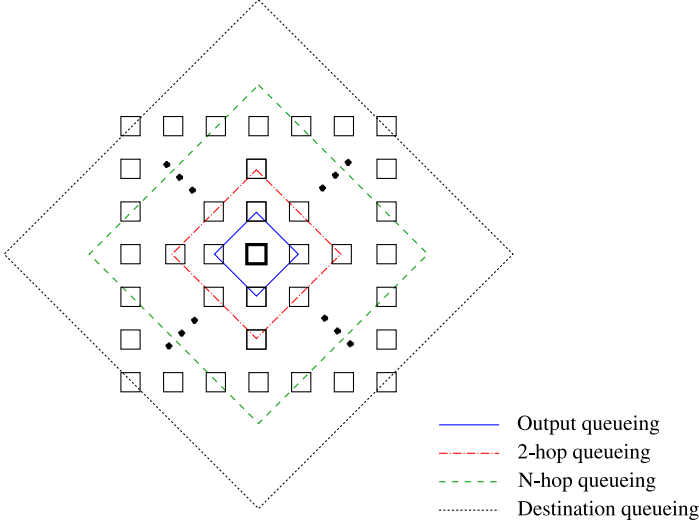


Figure 4.9: Regions of individually identifiable through-nodes from the perspective of a central node resulting in varying queuing strategies in a 49 node mesh network.

4. FLOWS, NOCS AND EFFICIENCY

carry some form of destination identifiers anyway. Therefore no additional information needs to be carried.

In a simple approach, the base-case router can be made non-interfering between flows by statically providing a single VC for each flow at each point in the network [20]. With continued technology scaling at the rate of Moore’s Law, the number of computation tiles, and hence the network size, on a chip doubles at each generation. Providing a unique VC for each destination therefore requires a doubling of the number of VCs in each router for every new technology node. Providing more VCs requires more resources and the tight on-chip power and area constraints means that a continuous increase in the number of VCs is entirely impractical. The solution of providing a unique VC for each destination does not scale to increasing numbers of nodes. A design where the power, performance and area cost of a single router (in this case set by the number of VCs) increases only slowly with increasing network size – a scalable solution – is required.

The provision of a VC for every flow possible at a point represents a design for the worst-case where every possible flow may be seen in a router at the same time. The novelty of this work is to enable flow support for a typical, rather than worst-case condition. In reality, the communication graphs seen in most future application classes discussed in Section 4.2 are of a low dimension [26; 49; 100]. More generally, it has been shown that using localised communications is the only way to achieve on-chip network scalability [34]. Typically, only a low number of flows might therefore be seen in a router at any one time. This strongly motivates the design for a typical case. With this approach, only a limited number of VCs (much lower than the total number of destinations in the system) need to be provided in each router, bringing down the cost to an acceptable level. However, since it is not possible to predict exactly which flows will use a router at a particular period in time, flows cannot be statically assigned to unique VCs. Instead, a mechanism must be developed where flows are dynamically assigned to VCs as they arrive at each router (Figure 4.10). Just as localised traffic limits the wire bandwidth demands to manageable levels, it also limits the requirements of the number of VCs, ensuring that the scheme scales.

The scalability of the design can be further investigated with observed localised traffic distributions. The localised traffic distribution empirically observed by Greenfield (the same as was used in the performance analysis in Section 3.7.2) was again used [34]. Equation 4.1 gives the relationship between hop-count and proportion of traffic going to that hop-count away observed in that study.

$$f = Ae^{-\alpha h} \tag{4.1}$$

where h is the hop count, f is the fraction of traffic going that many hops and A and α are constant parameters. Again taken from [34] A and α were set to 0.62 and 0.47 in this study.

Beyond this, a more relevant parameter is how much traffic goes *up to* a certain number of hops away. This can be combined with the topology and routing information to predict how many flows might be seen in a router in a typical case. Figure 4.11 shows a 2-D mesh topology, with a source node communicating up to 2, 3 and 4 hops away (only one quarter of the destinations are shown, as the analysis is symmetrical in all quarters). The highlighted link represents the highest loaded link with the triangular regions showing the number of destinations reached for a particular hop count limit. For hop count h the number of destinations is given by h^2 . For all these flows to be identifiable the number of VCs at that point must be greater than or equal to the number of destinations. This sets the limit on the number of VCs required for that link. With flows identified by destinations, considering additional source nodes does not change the number of flows seen at a link, as the number of destinations reachable from a link stays the same.

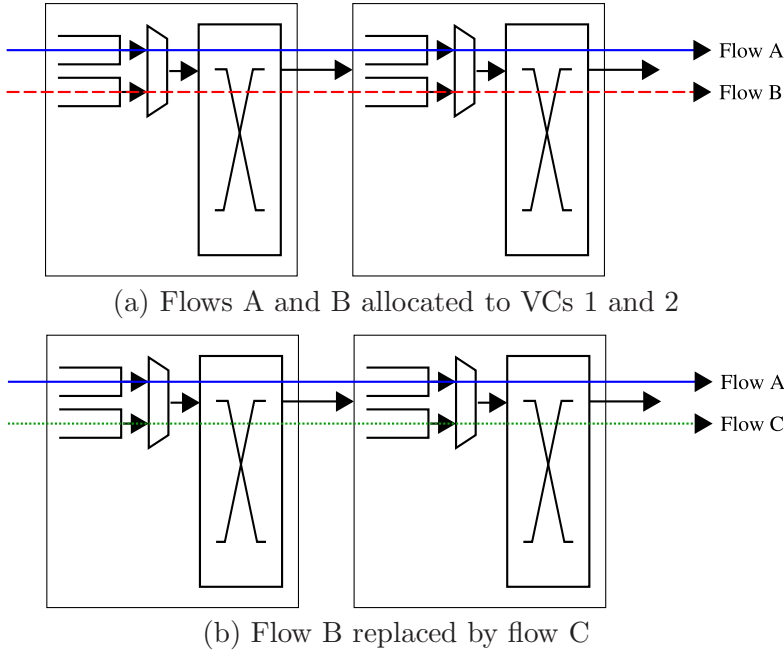


Figure 4.10: Dynamic allocation of flows to VCs. Instead of VCs being uniquely associated with individual flows, they are dynamically linked to arriving flows.

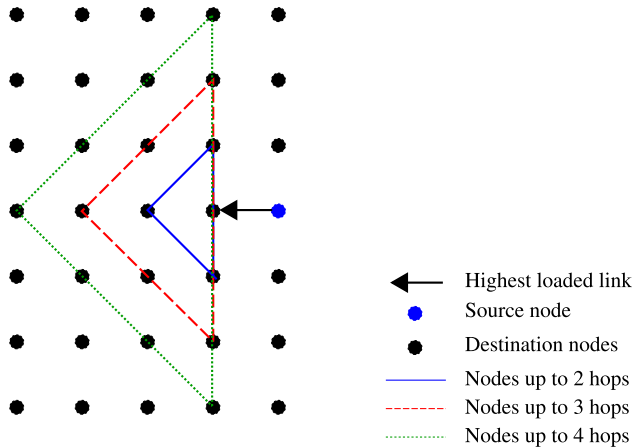


Figure 4.11: Number of destinations for up to 2, 3 and 4 hops away.

4. FLOWS, NOCS AND EFFICIENCY

Hop Count	Traffic fraction covered (%)	Number of VCs required for topology	
		Mesh	Flattened-butterfly with $l=2$
1	38.8	1	1
2	63.0	4	2
3	78.1	9	5
4	87.6	16	8
5	93.5	25	13

Table 4.1: Number of VCs required to efficiently schedule varying proportions of traffic going up to different hop counts.

Columns 1 (Hop Count) and 2 (Traffic fraction covered) in Table 4.1 list the first four hop distances and the fraction of total network traffic going up to that hop count distance given the traffic distribution of Equation 4.1. Columns 3 and 4 in the same table then show the number of VCs required to enable efficient scheduling for the associated traffic fraction for both a mesh and a higher order flattened-butterfly topology. As can be seen, with a mesh topology a reasonable 9 VCs is sufficient for efficient scheduling of nearly 80% of all on-chip traffic. Note that the remaining 20% of traffic is not dropped, but rather scheduled in the existing inefficient manner. This is an application of Amdahl’s Law of optimising the common case (in this case flows with hop-counts below a certain limit). With up to 16 VCs (as in the design presented by Nicopoulos [68]), efficient scheduling is enabled for nearly 90% of all traffic. The h^2 growth in the number of flows however does make efficiently scheduling more traffic increasingly expensive. But this problem is significantly reduced given the use of higher order topologies, such as the flattened-butterfly network proposed by Kim *et al.* [48]. As discussed in Section 2.2.1, other strong pressures for using higher-order topologies already exist and this work shows another benefit they can provide. With a flattened-butterfly topology with each router providing l links in each direction, the number of VCs required is closer h^2/l . Table 4.1 shows that even with l as low as 2, 13 VCs are sufficient to efficiently schedule more than 90% of all on-chip traffic.

4.3.4 Dynamically allocating flows to VCs

With flows allocated to single VCs in each router, only one packet from a flow must exist in a router input port at one time. This will enforce the ‘minimum resource usage’ policy discussed in Section 2.2.2. Packets of a flow should therefore pass through a router in a pipelined order as shown in Figure 4.12. For the implemented system, a table based approach has been designed to achieve this. A table (referred to from now on as the *flow table*) is provided at each output port with a row for each possible VC in the downstream router’s associated input port. Each row contains an *active* bit, and a *flow ID* field. Since in this work, flows are identified by their final destinations, the flow ID is equivalent to the destination address used. Given the 2-D mesh topology used here, the flow ID therefore becomes the X and Y destination co-ordinate identifiers used by the routing algorithm. When a head flit is allocated to the output link (by

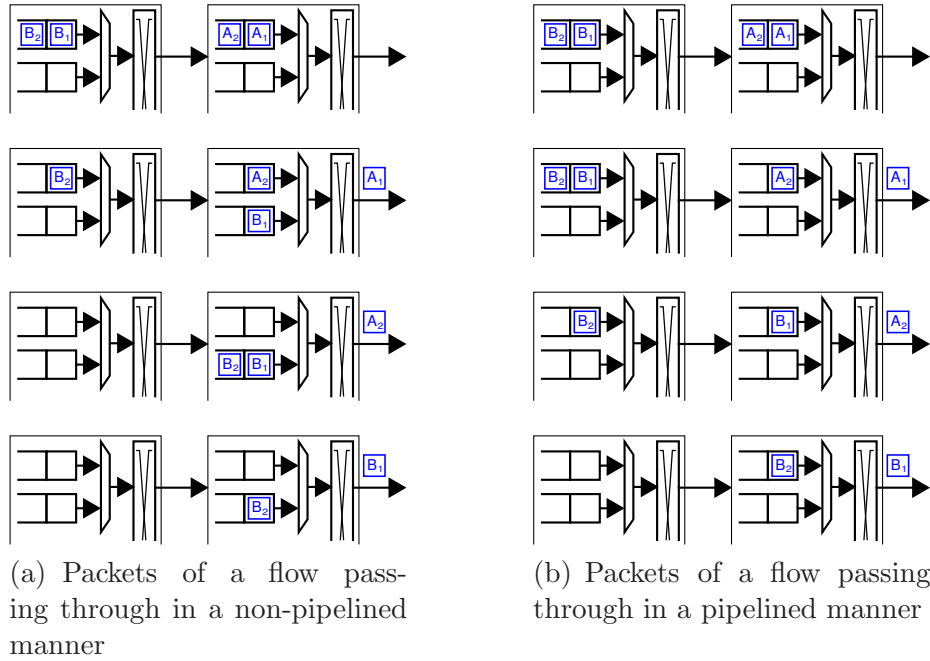


Figure 4.12: Packets of a flow passing through routers in a sequential, pipelined manner ensures that the minimum resource usage policy is enforced.

the switch allocator) the row identified by the packet's allocated VC has its active bit set and the packet's flow ID written to its flow ID field. As part of the allocation mechanism, head flits at the input VCs now search the table at their required output port for their flow ID stored in an active row. If a match occurs, that packet's switch request is blocked to prevent it from making forward progress. Successful allocation is therefore only achieved by a packet when no active entry is found in the flow table with a matching flow ID.

When the same packet discussed above leaves the downstream router its associated entry in this flow table must be invalidated to ensure that other packets of the same flow can now make progress. This can be efficiently added to the upstream signalling already used by credit-based buffer management. As discussed in Section 2.2.3, with credit-based buffer management every router keeps count of the number of free buffer spaces in the downstream router's input queues. Every time a flit leaves an input queue, a *credit* is sent back to the upstream router identifying that queue – i.e. the VC ID of the leaving flit. On receiving this signal the upstream router increments its count of free buffer spaces for that VC. Importantly, since all input VCs at a router input port share a single input into the crossbar, at most a single flit can leave a router's input port in 1 clock cycle. This in turn means that at most a single credit can be sent to the upstream router in a clock cycle. In a flow-based system therefore, only a single bit of upstream signalling needs to be added. When a credit is sent upstream, this additional *flow freed* signal can indicate whether the flit that caused the credit to be sent also signifies that a packet of a flow is leaving the router.

A naïve approach would be for every input queue to send this flow freed signal to the upstream router when the tail of a packet leaves. However, this would mean that the head flit of the next packet would not arrive until a round-trip-time (RTT) number of cycles later, thus wasting these cycles. However, since the RTT can be statically determined and given the use of wormhole switching, an optimisation is possible. Wormhole switching means that

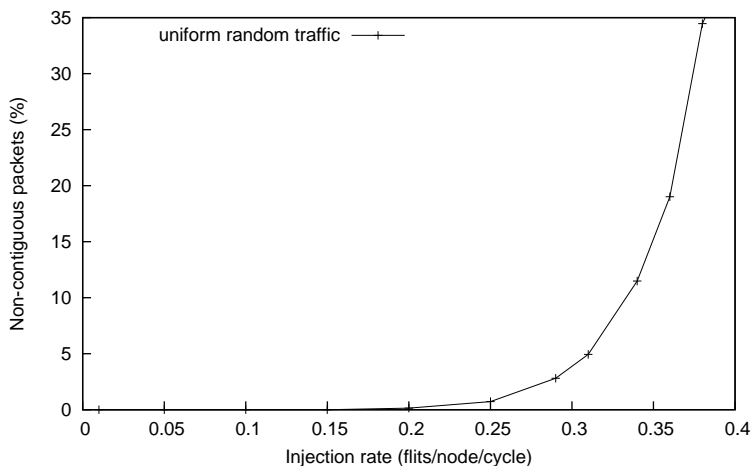


Figure 4.13: Percentage of packets sent non-contiguously for 8×8 network with uniform random traffic at varying injection rates.

packets are always sent contiguously unless they block halfway through. Such blocks rarely occur, especially in low traffic conditions. This is demonstrated in Figure 4.13. For the same uniform random traffic test results from Figure 4.4 earlier, this experiment measured the number of non-contiguously sent packets observed on the output ports of the router at co-ordinates (3, 3). As can be seen, even by the time the network saturates near an injection rate of 0.35 flits/node/cycle only around 17% of packets are sent non-contiguously.

Given this, it is quite accurate to optimistically assume that packets are always sent contiguously. This means that the flow freed signal can be sent when the flit that is an $RTT - 1$ number of places before the tail is transmitted. In the single cycle design used an RTT of 2 means that this signal is sent when the ($tail - 1$) flit is transmitted¹. If the packet actually blocked (i.e. the wormhole switching prediction was incorrect) the implication would be that the next packet of the flow could arrive, therefore using up a second VC and breaking the minimum resource usage policy. However, the network would still be functionally correct. Various approaches could be used to prevent this second packet from making further progress. Firstly, additional state could be added to the flow-table to block this second packet until the first one leaves. Alternatively, the switch allocator could be modified to grant the packets of the same flow on a first-come-first-served basis. This is the chosen approach of this thesis and is further discussed in Chapter 5. This chapter does not explicitly block this second packet noting that the presence of this second packet is unlikely in the typical case. The benefits observed in Section 4.3.5 even with this simplified scheme demonstrates the success of the typical-case design. Figure 4.14 illustrates this optimisation with Figure 4.15 showing the new VC allocator structure at each output port.

To provide full non-interference between flows it is important to ensure that flows do not conflict at any point in the communications infrastructure. This means that the flow-table mechanism must also be provided at the output of every computation tile's network interface. Two VCs are also required at the ejection point of the network to maintain low

¹Note that it is not necessary for packets to carry a count of their total number of flits to enable routers to calculate the correct 'flow freed' flit position. A single extra bit can be carried and used by the source to mark the position of the relevant penultimate flit in a packet.

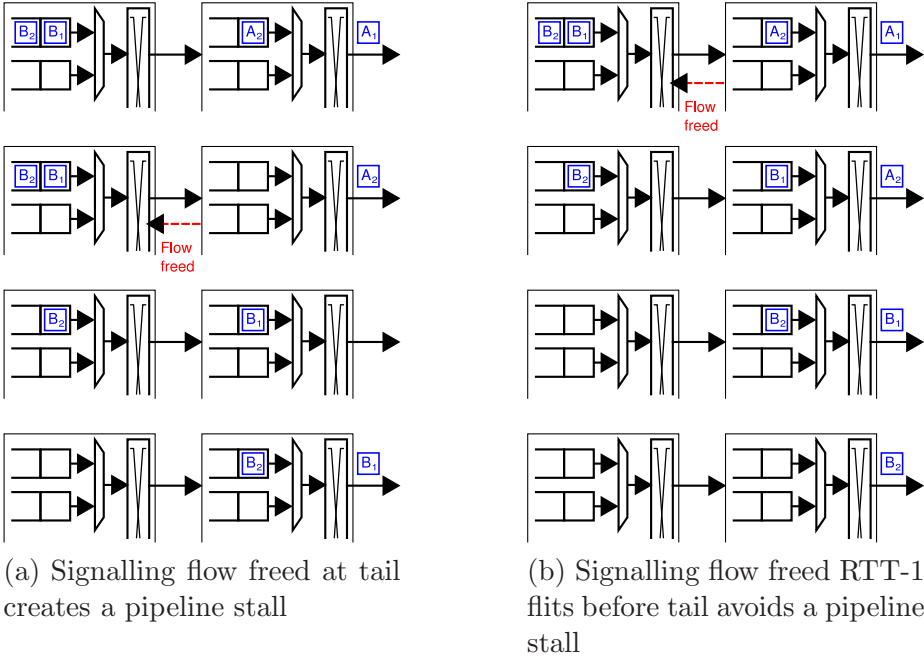


Figure 4.14: Optimistic signaling of flow freed event.

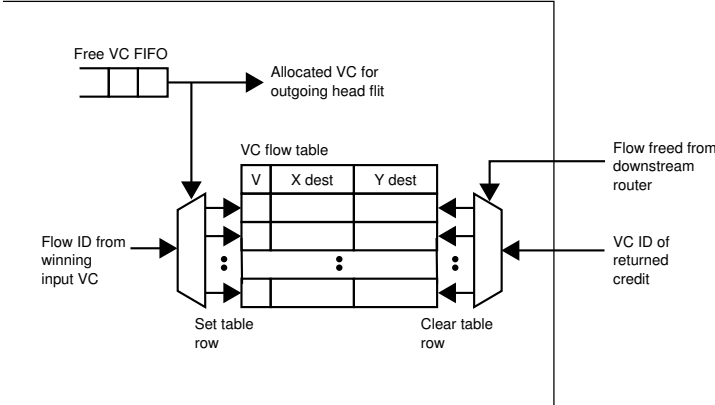


Figure 4.15: VC allocation logic, including VC flow table and mechanisms to set and clear it, at one output port.

4. FLOWS, NOCS AND EFFICIENCY

delay transmissions. If a single VC were provided, a delay of an RTT number of cycles would be incurred in the signalling of that VC being freed before the next packet could be transmitted. Providing a second VC avoids this delay.

With the table based mechanism described above it is important to realise that packets of a flow do not continuously re-use the same VC (as appears to be the case from Figure 4.10). The order in which the VCs are used is still set by the free VC FIFO (and is hence still LRU). Finally, in the presence of non-flow based communications, no flow dependencies exist for the flow-table to enforce and the VC allocation simply reverts back to the existing free VC FIFO based allocation.

The size of the flow table is an important factor determining the power and delay overheads of the design. For the 8×8 network considered here, 3-bit destination identifiers are used for each of the X and Y dimensions. With the additional ‘active’ bit the size of each row is 7 bits. For the 8 VC configuration used, this results in 56 bits per output port. This is still much smaller than the 128-bits and wider flits seen in recent NoC designs [50; 51]. The data-path’s power dominance over the control-path discussed in Chapter 3 therefore still applies. The approach of using more control to optimise the data-path is therefore again applicable. The more efficient data-path resource usage with this scheme means that fewer resources could be provided in the data-path to greatly reduce its power at the cost of slightly increased control-path power. Moreover, as discussed in Section 4.3.3 an important optimisation for scalability is to limit the number of flows supported at each node, given traffic locality. As discussed, this can limit the number of VCs required. Beyond this however, it can also limit the address widths used to identify flows. This is the same as limiting the width of the flow-table’s rows, improving scalability once again. As in Section 4.3.3, considering a setup supporting traffic up to 3-hops away, requires 9 VCs at each link. The associated 4-bits required to identify all supported destinations from each link then combine with the 1-bit active field to result in a table size of 45-bits at each output port.

Finally, the critical-path of the router can also be affected but several existing techniques of parallelisation or pre-computation of the allocation are applicable, which can minimise this overhead [50; 65].

The dynamic mapping of flows to a limited number of VCs presented here can be considered analogous to a cache memory with data items being mapped to a limited set of cache memory locations. The table-based allocation mechanism of this work can then be considered to be similar to a fully-associative caching strategy. Clearly other strategies such as the kind of hashing so common in caches could also be applied to flows and VCs. This is beyond the scope of this work but remains an important future direction.

Finally, the static traffic patterns originating from the presence of flows leads to severe fairness problems in bandwidth division, given that existing allocators are optimised only for bursty-traffic. All, or part of, the switch allocation modifications described in Chapter 5 are therefore necessary to fully enable support for flows.

4.3.5 Results

This section evaluates the flow efficiency properties of the proposed design added to the base-case router architecture. As in Section 4.3.1, we begin with the case of flows blocking due to congestion at a destination node by looking at a hot-spot traffic pattern. As in that section, an 8×8 network was used with uniform-random traffic, but with 5% of each source’s traffic going to a single hot-spot destination at co-ordinates (3,3), with 400 warm-up packets and 2000 measurement packets sent by each node. Each network link was once again divided into 8 VCs. The delay of both the hot-spot and non-hot-spot traffic is shown in Figure 4.16. As in

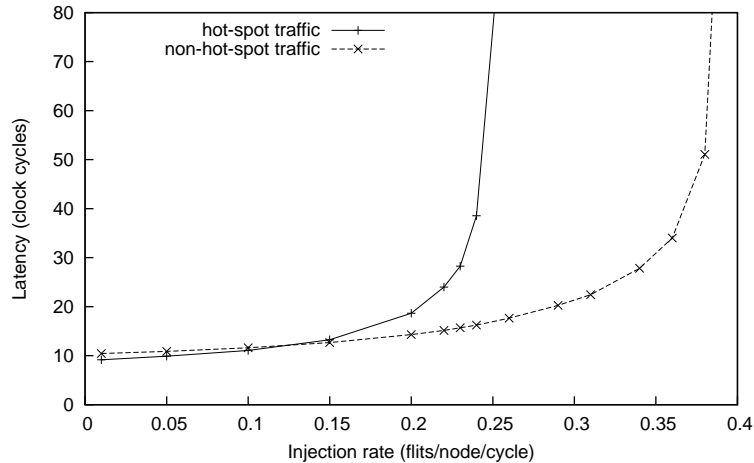


Figure 4.16: Hot-spot traffic latency versus injection rate with flow based dependencies enforced.

Figure 4.5 the hot-spot traffic saturates at an injection rate of around 0.24 flits/node/cycle. However, unlike in Figure 4.5, since these flows now do not take up all other network resources the rest of the traffic does not block and saturates near the uniform-random rate of 0.35 flits/node/cycle.

The second traffic scenario demonstrated in Section 4.3.1 was the case of flows needing to be limited due to congestion at a network link. A translation traffic pattern, considered to be an adaptation of the tornado traffic pattern, (shown in Figure 4.6) demonstrated very poor link utilisation due to flows taking up all network buffers upstream of a congested link. Figure 4.17 now shows the same experiment with the new design. As can be seen, enforcing the flow dependencies eliminates this problem as flows do not over-utilise network resources, leaving them free for other flows.

In the absence of traffic from real applications, additional synthetic traffic patterns were used to further evaluate the design. The classical bit-permutation traffic patterns described in Table 4.2 [23] were applied to the 8×8 network already presented. With each of the traffic patterns, every source was set to transmit 1000 packets to its corresponding destination. Table 4.3 then shows the number of clock-cycles required for every packet to be received for both the base-case and the modified architectures. As can be seen, not all traffic patterns exhibit flow-based inefficiency problems, showing no improvements with the flow-based allocation additions. However, where such problems exist the better resource utilisation provided by the new design directly translates to reduced time to completion, i.e. increased performance.

It is important that the flow based additions do not lower the performance seen for non-problematic traffic conditions. Table 4.3 has already shown that flow-based patterns that do not suffer from the inefficiency problems discussed are not slowed down by the new scheme. To test for non-flow-based traffic, Figure 4.18 shows the performance of the flow-based allocation scheme with the uniform-random traffic setup from Section 4.3.1. As can be seen, the flow-based allocation scheme does not limit non-flow based traffic in any way and therefore achieves the same performance as in Figure 4.4.

Finally, the flow patterns used here have all resulted in fewer flows than VCs at each network link. With the translation traffic experiment, for instance, up to 4 flows use a router input port, with 8 VCs at each link. This means that all the flows can be accommodated –

4. FLOWS, NOCS AND EFFICIENCY

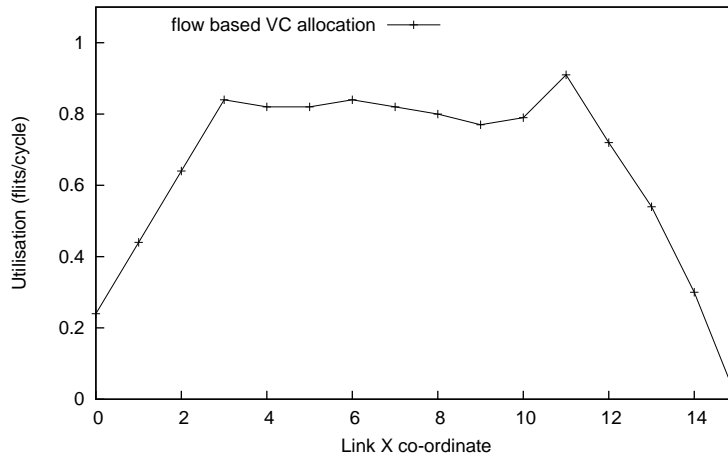


Figure 4.17: Link utilisation under translation traffic with flow based dependencies enforced

Pattern Name	Address function
Transpose	$d_i = s_{i+b/2 \text{ mod } b}$
Shuffle	$d_i = s_{i-1 \text{ mod } b}$
Bit rotation	$d_i = s_{i+1 \text{ mod } b}$
Bit reverse	$d_i = s_{b-i-1}$
Bit complement	$d_i = s_{-i}$

Table 4.2: Bit permutation traffic patterns. Each bit d_i of a b -bit destination address is a function of a single bit s_j of the source address, with j being a function of i .

Pattern Name	Completion time (clock cycles)		Speedup (%)
	Base-case	Flow-based	
Transpose	28038	28085	0.0
Shuffle	19402	18026	7.6
Bit rotation	22207	18148	22.4
Bit reverse	28038	28022	0.0
Bit complement	25907	16061	61.3

Table 4.3: Bit permutation traffic results.

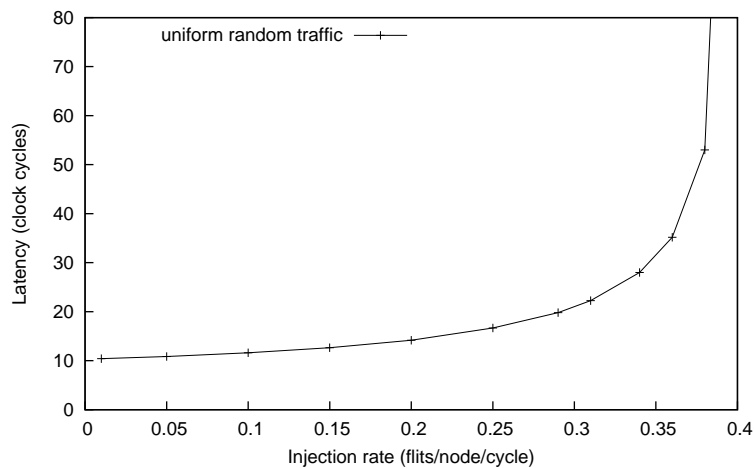


Figure 4.18: Uniform random traffic latency versus injection rate with flow based dependencies enforced.

i.e. the typical case design is able to cope with the traffic requirements. If the number of flows exceeds this typical case design, the performance is expected to drop. This is shown in Figure 4.19, where the same traffic pattern was used but the number of VCs reduced to 2. As expected, the high link utilisation cannot be sustained as flows cannot be distinguished given the limited number of VCs. This shows the necessity of performing a good cost-benefit analysis to correctly select the number of VCs for the network.

4.4 Summary

This chapter has shown that many expected future application classes exhibit much more stream-like communication *flows*. Achieving maximum system performance then demands that NoCs maximise communication performance in the presence of flows. Achieving maximum performance given limited power, wire and other constraints is then equivalent to maximising the resource-utilisation efficiency achieved by NoCs. The high efficiencies possible with dynamic scheduling mechanisms then justify the use of dynamic flow-aware scheduling in NoCs.

Current dynamic allocation based NoCs were shown to make highly inefficient resource usage with flows, with part of the problem being that no flow identification mechanisms are used. A framework was therefore developed where flows were identified according to their destination addresses. A table-based mechanism then dynamically mapped flows to network VCs with the restriction that, in a typical case, no more than one packet should exist in a router input port at the same time. Given localised traffic, it was shown that limits can be placed on the number of VCs required, importantly making the system scalable.

Analysis with various flow-based traffic patterns showed that the more efficient resource utilisation of the proposed system directly resulted in speedups of up to 61.3%. Moreover, the design was also shown to maintain high performance for non-flow-based traffic. The initial aim of maximising the performance for both flow and non-flow-based traffic given a fixed amount of resources is therefore well achieved. The approach introduced here of explicitly operating on units of flows further enables other important services to be provided by dynamic allocation based NoCs, such as the fairness mechanisms discussed in the next chapter.

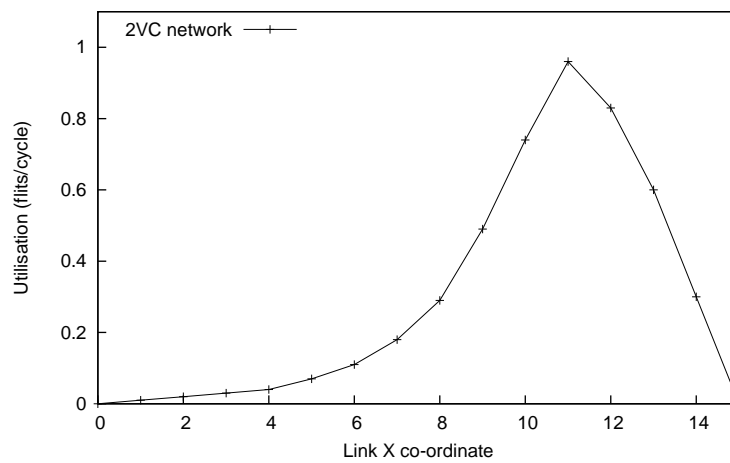


Figure 4.19: Link utilisation under translation traffic with flow based dependencies enforced but the number of flows exceeding the number of VCs.

Fair Allocation to Flows

5.1 Introduction

In any situation where a number of agents share a single physical resource, some form of *fair* division of that resource is essential. In most instances, a shared resource system can be said to have failed if some users receive a grossly unfair share of the service. Fairness questions are therefore of critical importance and they have featured strongly in fields as diverse as computation task-scheduling and economics.

Fairness issues have been especially important in the design of the various large-scale telecommunications systems in use today, especially the Internet. Several definitions of fairness, such as *max-min fairness* or *proportional fairness*, or fair allocation methods have therefore originated in this field. Due to this and the many conceptual similarities between large-scale telecommunication networks and NoCs, this field represents a good starting point for any NoC fairness research. Within communication networks, fairness questions also closely relate to those of *congestion control*. The ubiquitous nature of such mechanisms, such as the use of Transmission Control Protocol (TCP) on the Internet, therefore again point to the importance of these schemes.

Fairness questions tackled in the NoCs field to date have primarily focused on working with units of packets and not flows. As described in the Chapter 4, in the presence of flows, this approach leads to problems with resource utilisation efficiency. In a similar way, the same approach also makes it difficult to achieve fairness between flows.

Within any fairness-providing framework, it is a small step to change the relative importance of different users to provide different service levels to them, i.e. provide different QoS levels. The well investigated field of QoS support within NoCs therefore comes closest to attempting to allocate fairly to flows on-chip. As discussed in Chapter 4, the most common approach here has been to use a static division of the resources. As a result of this, the well known problems of poor resource utilisation efficiency and resource reservation delay of static allocation methods then again apply. The focus of this chapter is instead to investigate the use of efficient, dynamic NoC allocation techniques to achieve a fair division of resources across flows. A myriad of additional questions and alternative potential solutions are raised by this approach which lead to many possible future research directions.

5.2 The problem

Consider the situation demonstrated in Figure 5.1 where a single router has seven flows going through it, with each making the indicated bandwidth demand. If equal importance were placed on each of the flows, the bottleneck north output link should be shared equally between the different flows (such qualitative discussions of fairness are formalised in Section 5.3). In a flow-based fair system, each flow would therefore be expected to receive around 14.3% of the link's bandwidth.

5. FAIR ALLOCATION TO FLOWS

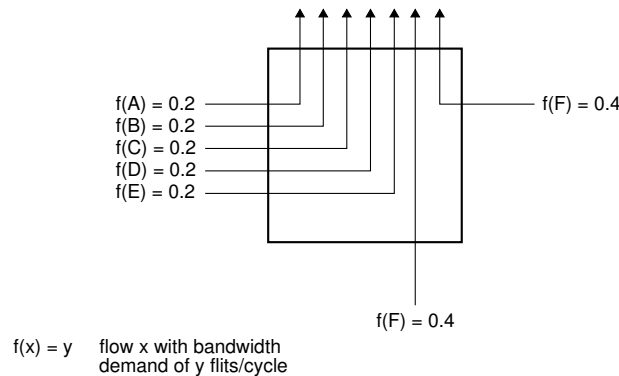


Figure 5.1: Seven flows passing through a single router, with associated bandwidth demands shown in flits/cycle.

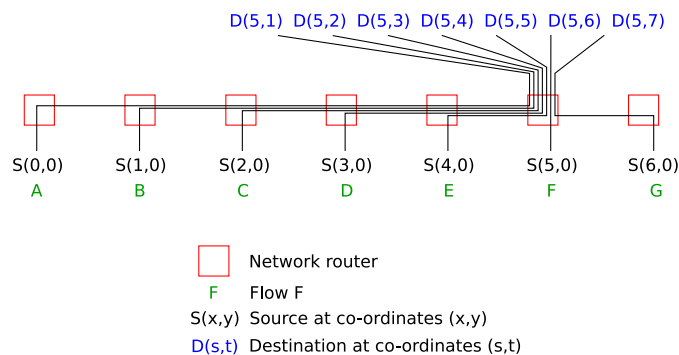


Figure 5.2: Network setup to evaluate unfair allocations.

To model this scenario, the base-case router was tested with the network arrangement and flow pattern shown in Figure 5.2. The source router of every active flow injected 400 warm-up packets, followed by 2000 measurement packets, which were the only ones used for any measurements. All active routers then continued to inject packets at their set rates until all measurement packets were received. Column two (labelled base-case) in Table 5.1 shows the resulting bandwidth allocated to each flow.

As can be seen, the achieved rates are far from matching the desired rates. In any general purpose computing environment, with many independent threads of computation generating flows, this would result in some threads being starved out. The vastly differing performance seen even across multiple similar applications would make this unacceptable to the user. A large gap will exist between the worst and best-case performance. Over-provisioning of network resources may then be necessary to ensure that the worst-case is still acceptable. Such over-provisioning is inefficient. More resources have to be deployed than that required by a fair allocation scheme. Because of this, current NoCs already place a strong emphasis on fairness. However, as will be discussed, this is done on a per-packet and not per-flow basis. Given the importance of flows in many emerging applications, the aim of this chapter is to enable a fair division of resources between flows in a NoC.

Flow	Allocated rates (flits/cycle)	
	Base-case	DF router
A	0.02	0.07
B	0.03	0.07
C	0.04	0.07
D	0.08	0.07
E	0.17	0.07
F	0.33	0.33
G	0.33	0.33

Table 5.1: Unfair flow allocations in base-case and DF routers.

A number of design flaws lead to the unfair allocation result. As already discussed in Chapter 4, the first problem is that network routers do not have any means by which to identify different flows. As described in Chapter 2, the basic unit of allocation in the base-case router is a single virtual-channel (VC), occupied by individual packets. No flow-based information is used to allocate packets to VCs and hence nothing can be inferred about the presence or absence of flows at any of the routers' input ports. Clearly, if flows cannot be differentiated, no allocation mechanism can be developed to fairly arbitrate between them. The flow identification mechanism developed in Chapter 4 therefore represents the critical first step for any fairness supporting scheme. In that chapter, a flow is identified as the group of all packets going to the same destination – referred to from now on as a *destination-flow*. The router design subsequently developed (referred to from now on as the DF router) then ensures that, in a typical case with scalable localised traffic, only a single packet from a flow exists in a single router input port at any one time. This simple modification leads to the important ability of a single input VC to represent a single flow. Outside of the typical case, the functionality reverts back to the base-case design.

For the same experiment as described in Figure 5.2, column three of Table 5.1 (labelled DF router) shows the allocated bandwidth to the different flows achieved by the DF router. As can be seen, even though flows are now identifiable, a fair division across flows is not achieved. The equal rates allocated to flows A to E is caused by the same mechanism as that which solves the translation traffic inefficiencies described in Chapter 4.

The unfairness here arises from the use of separable allocators to allocate bandwidth within routers. With separable allocators, router output ports effectively group together incoming requests on a per-input-port basis. The output arbiters then arbitrate across these groups, ignoring the available flow presence information at the inputs, as shown in Figure 5.3. This lack of use of the flow information fundamentally restricts the ability to allocate on a per-flow basis. In the same way as shown by the previous chapter of VC allocation needing to occur on a per-flow basis to ensure efficient resource utilisation, providing fair bandwidth division across flows requires that the switch allocator also operate on a per-flow and not a per-packet or per-port basis.

Additionally, consider the traffic pattern shown in Figure 5.4, where 5 sources are sending traffic to the same destination. The equal importance previously placed on all destination nodes demands that equal importance also be placed on all source nodes. Each source should

5. FAIR ALLOCATION TO FLOWS

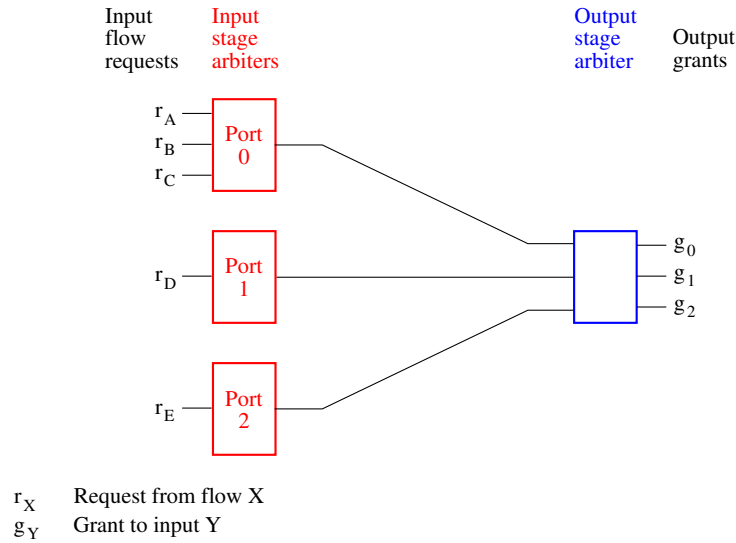


Figure 5.3: Unfair separable allocator does not account for varying number of flows at different input ports contending for the same output port.

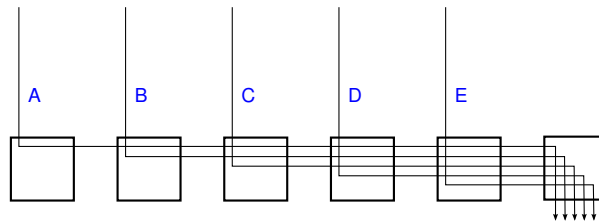


Figure 5.4: Traffic pattern with five sources sending data to a single destination node.

therefore receive 20% of the bandwidth at the destination node. Table 5.2 shows that this is not the case for both the base-case and DF router configurations. With this setup, even the DF router cannot achieve a fair division across flows due to the nature in which it identifies flows. Since flows are only identified by destination addresses, no ability exists to separate traffic coming from different sources. Any flow fairness supporting scheme not only needs to operate on a per-flow basis, but sufficient information needs to be present in the system to allow flows to be identified at the correct level of granularity.

5.3 Background and related work

In a communications network with a particular flow-based traffic pattern, the possibility exists of the traffic demands being low enough that all network elements are under-subscribed. Clearly, all flow demands can then be met and no unfairness will exist. However, this strict under-utilisation condition cannot always be guaranteed. Especially in the tightly constrained on-chip environment, a strong pressure exists to prevent simple over-provisioning of resources. Instead, it is much more desirable to make good use of a reasonable amount of a resource. Mechanisms to enable fair division of that resource will then be necessary for the inevitable

Flow	Allocated rates (flits/cycle)	
	Base-case	DF router
A	0.06	0.06
B	0.06	0.06
C	0.13	0.13
D	0.25	0.25
E	0.50	0.50

Table 5.2: Unfair flow allocations in base-case and DF router for hot-spot traffic.

periods when the total demand exceeds the available supply.

To aid further progress Shenker introduced the concept of a *utility function* for each user [84]. For a set of users \mathcal{S} , the utility function $U_s(x_s)$ of each user $s \in \mathcal{S}$ represents the value placed by source s on being allocated a service rate of x_s . In the case of networks, the service represents the supported traffic rate. Given additive utilities, the goal of any network allocation policy can now be concisely stated as the maximisation of the total of all the users' utilities.

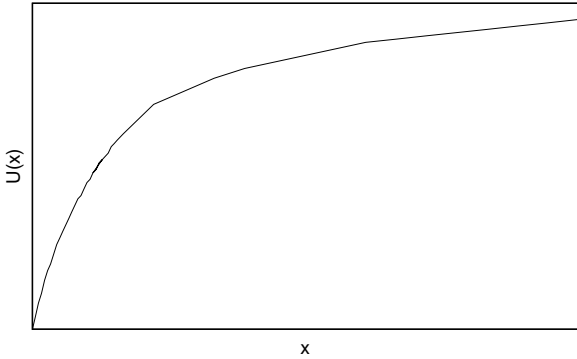
$$\sum_{s \in \mathcal{S}} U_s(x_s) \quad (5.1)$$

The shapes of these utility curves have an important impact on many aspects of network design. Figure 5.5(a) shows the shape of the first class of applications introduced by Shenker [84]. Labelled as *elastic* applications, such utility function are *strictly concave*. Their shapes intuitively make sense – very little benefit is observed at low service rates, but beyond some point increasing service levels only return marginal overall gains. On-chip this could, for instance, be observed in the case of the scientific applications described in Section 4.2, where the communication limits performance at low service levels but beyond some point other limits begin to dominate.

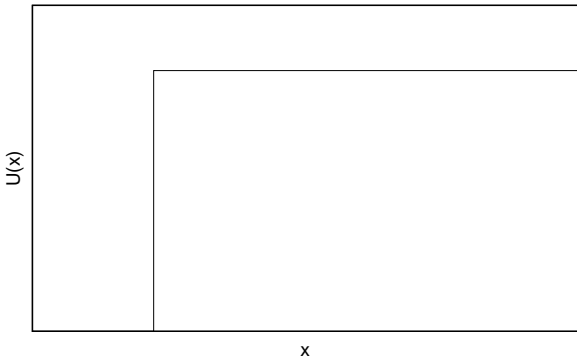
Two other important classes of utility functions, again introduced by Shenker, are shown in Figures 5.5(b) and 5.5(c). Figure 5.5(b) shows the utility function of a hard real-time application, where below an absolute minimum rate the application cannot even be run. A much more realistic real-time application scenario is shown in Figure 5.5(c), where applications can adapt their usage requirements according to network conditions. A basic traffic rate is still required but once this has been achieved the application can adapt to varying network conditions and do useful work even without achieving the maximum service level.

Outside of the specific topic of QoS research, fairness mechanisms in large-scale networks have so far primarily been based on elastic traffic. As with the scientific applications discussed above, the class of applications expected within the increasingly dynamic and parallelised on-chip environment again means that the same kind of traffic becomes important. Moreover, for the adaptive real-time utility functions, it is feasible to additionally implement some form of admission control to turn away new flows in a situation where allowing them to enter would lower the total utility. All flows will then operate in their concave region. This work therefore focuses on these strictly concave utility functions.

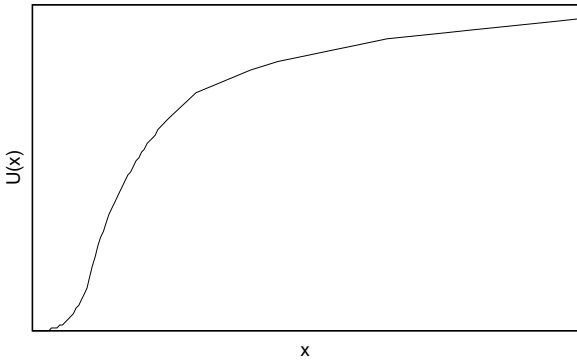
Given this groundwork, the problem of network elements becoming over-subscribed can



(a) Elastic traffic



(b) Hard real-time traffic



(c) Delay/rate adaptive real-time traffic

Figure 5.5: Different traffic utility functions.

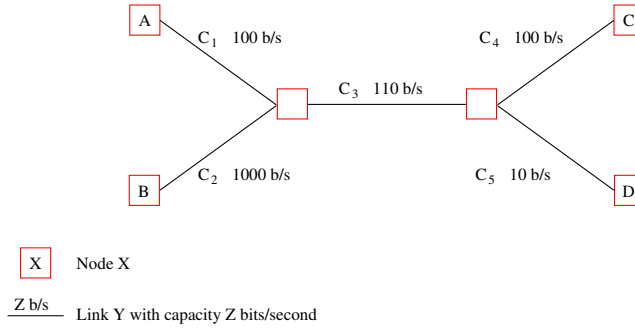


Figure 5.6: An example network setup demonstrating congestion collapse.

now be considered. First, consider an example adapted from [14] and illustrated in Figure 5.6. Within this setup, assume that sources only limit their injection rate according to the capacity of the network link they directly connect to. Also assume that network routers drop packets if the demand outstrips the output capacity, and input requests are served in proportion to their request rates. With sources A and B sending to destinations C and D respectively, a good allocation would limit source B to 10 bits/second and allow source A to utilise the full 100 bits/second at destination D. However it is easy to establish that, with the above setup, both sources only achieve a useful sink rate of 10 bits/second. This is the well known problem of *congestion collapse*. In NoCs, the inefficient allocation observed under translation traffic in Section 4.3.1 can be attributed to a very similar situation within VC routers. Here, flows do not limit their injection rates according to conditions on their route and one of the parameters network routers base their allocations on are the relative ratios of the incoming flow rates. The conclusion for large scale networks is the same as it must be for NoCs:

Proposition 1 *Sources must limit their injection rates according to conditions on their route [14].*

Within NoCs, the work of the previous chapter enables exactly this. With flows mapped to single VCs, the node-to-node back-pressure limits a source's injection rate to the lowest of the allocated rates to that flow's VCs along its entire path. This therefore forms the starting point for any NoC fairness research.

To allow a fair division of resources, the relative importance placed on different flows must first be decided. If this information is not available, the best that can be achieved is to assume that every flow is as important as every other (i.e. all users are assumed to have an identical utility function)¹. Moreover, the concave nature of the utility functions discussed above means that when a new flow arrives the total utility will always be increased by allowing that new flow to enter the network, regardless of the number of existing flows in the system. These two conditions then naturally lead to the classic definition of *max-min* fairness where, as far as possible, all resources are divided equally between all users.

Before progressing further, a network model is first presented. Let \mathcal{L} be the set of all links in the network, with each link $l \in \mathcal{L}$ having a capacity of $C_l > 0$. Let \mathcal{R} denote the set of routes using the network with each route $r \in \mathcal{R}$ being associated with a particular source and using a subset of \mathcal{L} . Finally, assume a fluid-flow model and let λ_r denote the rate allocated

¹Note that a form of QoS can now be easily supported by placing different relative importance on different flows.

5. FAIR ALLOCATION TO FLOWS

to route r . A feasible allocation can then be defined as one that satisfies all the capacity constraints:

$$\sum_{r \ni l} \lambda_r \leq C_l, l \in \mathcal{L} \quad (5.2)$$

Max-min fairness is then defined as that allocation which maximises the minimum λ_r , while still remaining in the feasible set – hence giving rise to the term max-min. It attempts to give an equal share to every flow, but if capacity constraints do not permit this, the route allocated the minimum rate is given maximum priority. Another representation therefore becomes that the max-min fair allocations λ_r are such that any increase in a particular λ_r can only be obtained at the expense of a decrease in $\lambda_{r'}$ for any other route $r' \in \mathcal{R}$ where $\lambda_{r'} < \lambda_r$. It can be shown that an allocation will be max-min if every flow has at least one *bottleneck* link, with a bottleneck link being defined as below¹ [14].

Definition 1 A *bottleneck link* for route $r \in \mathcal{R}$ is that link $l \in r$, $l \in \mathcal{L}$ where:

- 1 $\sum_{r \ni l} \lambda_r = C_l$, i.e. the link is at full capacity.
- 2 $\lambda_r \geq \lambda_{r'}$, $r' \ni l$, $r' \in \mathcal{R}$ i.e. the allocated rate to that flow is greater than or equal to the rate allocated to any other flow r' using that link.

The particular method of *progressive filling* presented by Bertsekas and Gallager provides an intuitive method to achieve a max-min fair allocation for a *fluid-flow* model [10]. The algorithm is divided into distinct time-steps, each identified by the variable k . The basic idea is to start with all flows allocated a rate of 0 and increase all the allocations at the same rate until a capacity constraint is reached. At this point, the sources affected by this constraint keep their allocated rates, whereas all others again have their rates increased. This process continues until it is not possible to continue any further, giving the final allocation. Adapted from [10] and more formally, let \mathcal{L}^k represent the set of under-subscribed links and \mathcal{R}^k the set of routes not passing through any over-subscribed links at the start of time step k . Define n_l^k as the number of routes using link l that are in \mathcal{R}^k (i.e. the number of routes that can still have their rates increased). Define Δ^k as the increment of the rates and λ_r^k as the rate allocated to route r at step k . Finally, represent the used capacity of link l at step k as:

$$F_l^k = \sum_{r \ni l} \lambda_r^k \quad (5.3)$$

Initially start with $k = 1$, $F_l^0 = 0$, $\lambda_r^0 = 0$, $\mathcal{L}^1 = \mathcal{L}$ and $\mathcal{R}^1 = \mathcal{R}$.

- 1 $n_l^k =$ number of routes $r \in \mathcal{R}^k$ and $r \ni l$
- 2 $\Delta^k = \min_{l \in \mathcal{L}^k} (C_l - F_l^k) / n_l^k$
- 3 $\lambda_r^k = \begin{cases} \lambda_r^{k-1} + \Delta^k & \text{for } r \in \mathcal{R}^k \\ \lambda_r^{k-1} & \text{otherwise} \end{cases}$
- 4 $F_l^k = \sum_{r \ni l} \lambda_r^k$
- 5 $\mathcal{L}^{k+1} = \{l | C_l - F_l^k > 0\}$

¹Note that this is different to the usual meaning of the term *bottleneck*.

6 $\mathcal{R}^{k+1} = \{r | l \notin r, \text{ for all } l \in \mathcal{L}^{k+1}\}$

7 $k = k + 1$

8 If \mathcal{R}^k empty, then stop; else go to 1.

Note that, although not described above, it is elementary to modify this algorithm to account for flows reaching their demanded service level before encountering a saturated link by appropriately removing them from the set \mathcal{R}^k .

Once a fairness model and algorithm to achieve it have been identified the most obvious technique to implement it would be to centrally and statically calculate the fair source injection rates for a given traffic pattern and limit sources to this level (for an example see [2]). However, the inflexibility of static allocation mechanisms make them fundamentally unsuitable in the presence of any unpredictability. On-chip, the same arguments of increased efficiency from dynamic resource allocation as described in Chapter 4 justify the use of dynamic techniques to achieve fairness. The broad class of existing dynamic techniques to achieve fairness in a realistic packet-based (and not fluid-based) model are briefly described below.

Hahne was one of the first to demonstrate a dynamic, distributed technique to achieve max-min fairness [35]. For a repeating traffic pattern, perhaps with jitter, (i.e. analogous to the kind of behaviour of flows as discussed in Section 4.2) a max-min rate allocation can be achieved if every link services requests for that link in a round-robin fashion. However, this is only achieved if a flow always has a packet queued at a node when its turn comes up within the round-robin order. Hahne went on to show that with the use of node-to-node window based flow control (i.e. like credit-based flow control in NoCs) a packet can indeed be guaranteed to be available at a node when its turn comes up, given a minimum window size. In the worst-case unfortunately, the required window sizes can be un-feasibly large. The other primary limitation of this work is that fairness is only achieved for equal sized packets, given the packet-based arbitration scheme. The most important contribution of the work is perhaps that the mechanism is completely distributed with each node making decisions based solely on information present locally. Besides control for the window flow control mechanism, no additional communications are also required.

A very different technique to achieve fairness in a distributed manner has been proposed by Demers *et al.* [25] and Parekh *et al.* [70; 71] who started with the *Generalized Processor Sharing* (GPS) approach. GPS assumes a fluid-flow model where independent servers (i.e. network nodes) service flows according to a set of pre-defined weights. With equal weights for all flows, the GPS servers can be shown to achieve max-min fairness [25]. It has been shown that a simple modification can transform the fluid-based model into a more realistic packet-based model with little deviation from max-min fairness. If T_p represents the time at which a packet p would leave a network node, given GPS scheduling, a result close to GPS is achieved if that node services packets in increasing order of T_p . This scheme is known variously as *Packet Generalized Processor Sharing* (PGPS), *packet fair-queueing* or simply *fair-queueing*.

More recently Kelly initiated the field of Network Utility Maximisation (NUM) with the seminal paper on rate control for elastic traffic [47]. Kelly demonstrated that the maximisation of total system utility can be formulated as a single constrained optimisation problem for the whole system. Importantly, for strictly concave utility functions of individual users, this problem can then be decomposed into a local problem for each source and one for the network (which can be further divided into one for each network node). The various problems are linked by coupling variables between the sources and the network links those sources use. It was then shown that an iterative process can be used to converge to the global optimum. Moreover, at each step of the iteration each source and each network node need only alter

5. FAIR ALLOCATION TO FLOWS

local variables (such as injection rate for sources or link cost for network links). Kelly goes on to propose *proportional fairness* which reduces the emphasis on the smallest rate flow, inherent in max-min fairness. The utility functions $U_s(x_s)$ are then given by $\log(x_s)$. Mo and Walrand go on to show how existing fairness criteria fit in perfectly within the NUM framework by developing a set of utility functions parameterised by a variable α [60]. Different values of α , giving different utility functions, result in the system optimum representing different fairness criteria. The resulting fairness criteria are termed α -*fair* allocation schemes (shown in equation 5.4). Many important benefits are achieved with this framework. One important result shows that, since the utility function of all such fairness criteria are strictly concave a single global optimum exists.

$$U_s(x_s) = \begin{cases} (1 - \alpha)x_i^{1-\alpha} & \text{if } \alpha \neq 1 \\ \log(x_s) & \text{if } \alpha = 1 \end{cases}$$

$$\begin{aligned} \alpha = 0 : & \quad \textit{maximum throughput} \\ \alpha = 1 : & \quad \textit{proportional fairness} \\ \alpha = 2 : & \quad \textit{minimum potential delay} \\ \alpha = \infty : & \quad \textit{max - min fairness} \end{aligned} \tag{5.4}$$

Most existing research into fairly allocating to flows within NoCs has approached this problem from the indirect perspective of QoS issues. The example network designs mentioned in Section 4.2 therefore represent relevant fairness research in NoCs. Beyond this, Lee *et al.* have recently proposed a novel QoS service providing mechanism for NoCs which can in-fact be considered to be providing an approximation to max-min fairness [52]. With this scheme, network time is divided into fixed length *frames*. The network then attempts to equally distribute resources across all packets injected into the network during the same time frame. With such temporal schemes, however, some form of global synchronisation is required (in [52] a global barrier network is used) which limits the scalability of the design.

5.4 Approach for NoCs

As highlighted in Section 5.2, before any fairness mechanisms can be implemented it is important to select the communication entities between which fairness is to be achieved. In other words, the term *flow* needs to be better defined. Historically a variety of entities have been considered as flows. These have ranged from the use of physical source-destination address pairs as proposed by Demers *et al.* [25], to the more traditional use of application classes in terms of TCP/IP address/port pairs [82], to explicit flow labels [78] or even aggregation of other flow identifiers to result in separate flows [89].

Section 4.3.2 has already explained that the framework developed in this thesis is based around an existing, simple hardware-software interface. With this, packets injected into the network do not contain explicit flow-labels but only specify a network destination node. This chapter again assumes the same interface. However, just as in Chapter 4, the work presented in this chapter can be easily extended to operate with predefined flow-labels.

The solution to the efficiency problems of Chapter 4 required flows to be identified solely by their destination addresses. However, it can be concluded from Section 5.2 that for fairness, identifying flows by source-destination address pairs is much more meaningful. The rest of the work therefore takes this approach – with the resulting flows termed *source-destination-flows*.

The absence of any more complex hardware-software interfaces additionally means that the best assumption the hardware can make is to place equal importance on all flows. This

naturally leads to the desire for max-min fairness. The fundamental limitation of long wire delay (leading to the use of NoCs in the first place) moreover demands an entirely distributed (as opposed to a centralised) implementation of this. The power costs of data transfer demonstrated in Chapter 3 further demand that any additional information transfer is kept to a minimum. However, from the same chapter it is also known that a certain freedom is available in increasing the complexity of the control within each router. Separately, the not entirely predictable nature of on-chip communication demands means that any schemes requiring explicit knowledge of the desired flow rates are unlikely to be good candidates. Finally, as already mentioned in Section 4.2.3, although flows are important they certainly cannot be guaranteed to be the only type of traffic ever seen. It is therefore important to maintain good performance for non-flow-based traffic as achieved by existing NoC designs. Successful implementation of these guidelines would result in a general-purpose NoC, able to provide good performance to a wide range of traffic patterns, enabling maximum system performance to be achieved.

Despite the flexibility provided by NUM allocation mechanisms, they cannot easily achieve all the aims discussed above. Primarily, the inherent iterative nature of such solutions does not map well to non-flow-based traffic patterns. As a result, these might experience grossly unfair allocations and high network latencies. Existing methods to support non-flow-based traffic do not map well to on-chip environments, as most require solving the difficult challenge of identifying flow and non-flow-based traffic or necessitate complex heuristics at the sources.

Within a router implementing PGPS, the virtual finish times necessary in the allocation process are a function of the arrival properties of all input packets contending for the same resource. Although a calculation of this nature is feasible for IP routers, a low-cost on-chip implementation is challenging.

The distributed round-robin arbiters presented by Hahne [35] and discussed in Section 5.3, therefore appear most amenable to an implementation in NoCs. Very similar allocators are already used within NoC routers. Moreover, with delay being an important metric on-chip, these have been developed to achieve low-delay allocation. It appears feasible, therefore, to modify them to enable support for flows at low cost. Moreover, in the absence of flows the system can be made to revert back into an existing NoC implementation – maintaining good support for non-flow-based traffic. Beyond the ability to identify source-destination flows, no extra communication is also needed. Finally, no part of the computation requires a user’s demanded flow rate to be known. No need therefore arises to explicitly evaluate the demanded rates.

However, the problem of large window sizes required in the worst-case needs to be considered. As discussed in Section 5.3, with inputs being served in a round-robin fashion, max-min allocation is only achieved if a flow has a packet waiting at a node when its turn to be served comes up within that round-robin grant order. If no packets of a flow are waiting, it will not be served until another full round of allocations has completed. Hahne has shown that this need to have a packet waiting can be fulfilled by the use of large windows. Within NoCs, Chapter 3 has shown that buffers demand the most power. Due to this, they certainly cannot be greatly expanded to satisfy the large window demand. Instead, the use of matrix-arbiters (to give a *Least Recently Used* grant ordering) is proposed. These can ensure that packets in a router do not have to wait for an entire round of allocations, but are instead served at their max-min level at the earliest possible time after their arrival. At the expense of holding more state (i.e. increasing the control path complexity) they remove the need to use very large buffers (i.e. keep the data-path simple). As discussed in Chapter 2, low-cost implementations of matrix arbiters are already prevalent in NoCs. Moreover, the typical-case design approach of this work also limits the total number of flows (by providing a limited number of VCs) to

further limit the size and hence cost of the arbiters.

The other key disadvantage of the method proposed by Hahne is that max-min fairness is only achieved for fixed-sized packets. The typical case approach again helps, as most on-chip communications are expected to be of the same packet size (set by external parameters, such as the cache-line width). Other solutions, such as breaking up larger packets into fixed size packets, or working at the granularity of single or double flits are also possible.

5.5 Identifying source-destination flows

This section begins the development of a framework to achieve max-min fairness between flows in NoCs. As in Chapter 4, the aim is to develop a framework that can be applied to a variety of NoC architectures, as opposed to being tied to a single design. Again as in Chapter 4, the design cost is limited by providing additional benefits only for the typical-case of localised traffic scenarios. Throughout the rest of the chapter, the important delay overheads of the major additional components required by this scheme are analysed. This was performed by developing RTL models of the components and performing a topologically-aware synthesis (to account for the placement and internal wiring parasitic effects) of these models in a contemporary 90nm library.

In current virtual-channel NoCs, the VCs represent the basic unit of traffic identification for allocators within each router. Traffic entities are dynamically mapped to VCs and the allocators divide resources between active VCs. As discussed in Section 4.3.3, increasing the number of VCs results in increasing control-path costs. This provides an impetus to keep the number of VCs to a reasonable level.

Within the context of this work – of flows being dynamically mapped to VCs – Chapter 4 provides a strong motivation to identify flows by packets’ destination addresses. This enables good resource utilisation while minimising control overheads. However, it can be concluded from Section 5.4 that, from a fairness perspective, flows should instead be identified by source-destination address pairs. Mapping such source-destination-flows to VCs would allow VCs to represent source-destination flows, thus enabling the switch allocators to work at this desired level of granularity. However, a side-effect of this would be an increase in the number of VCs required to support the same amount of localised traffic. This, along with the additional complexity of dynamically mapping flows to VCs, makes the scheme of mapping source-destination flows to VCs undesirable.

Before a solution is proposed, consider the example traffic pattern shown in Figure 5.7, where two source-destination flows, going to the same destination (i.e. belonging to the same destination flow), intersect at node A. If VCs were allocated to source-destination flows, Figure 5.8(a) shows the sequence of events that would occur when packets from the two source-destination flows contend for router A’s output link at the same time. One of the packets, say packet S_0^0 , would first win the allocation and be forwarded to node B, using that source-destination flow’s allocated downstream VC. In the next time-slot packet S_1^0 would follow and use its own allocated VC. If however, flows were identified solely by their destinations, as in Figure 5.8(b), packet S_1^0 would instead be halted until packet S_0^0 left router B and then come in with the same flow ID as packet S_0^0 . Moreover, if router A’s arbiter serves source-destination flows in a least-recently-used manner, packet S_1^0 is guaranteed to be sent before any other packet from source 0, S_0^x is sent. This observation leads to the proposed solution of adding a *source-count* field to every packet. Set to 0 at the injection point of the network, this value can be incremented every time two or more packets going to the same destination coincide in a router, as in the case of router A in Figure 5.7. When a downstream router, such as B in Figure 5.7, sees a packet arrive with a source-count greater than 0, it

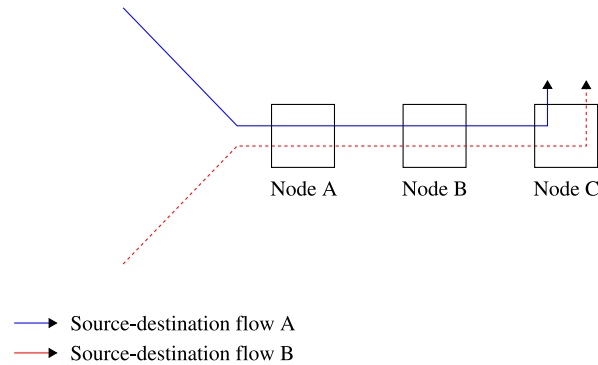


Figure 5.7: Two source-destination flows going to the same destination intersecting at a network node.

is informed that the next packet to arrive for the same destination will be from a different source. This results in the final proposed system configuration where VCs are still allocated by destination addresses (to give good resource utilisation with minimal control overhead, exactly as in Chapter 4) but source-count values are carried by packets to enable the switch allocator to identify individual source-destination flows. The exact mechanism in which this is used to achieve global max-min fairness will be described in Section 5.6.

The width of these source-counts clearly represents a critical parameter affecting the overall design costs. Importantly, they represent the additional communication necessary to identify source-destination flows. The typical-case design approach followed throughout therefore again applies. The low dimension communication graphs observed in on-chip, flow-based traffic patterns [26; 49; 100] allow a maximum width to be set for this. The rest of this work uses 3-bit source counts per packet to allow up to 8 sources to be identified per flow. Ideally, the quantitative approach of Section 4.3.3 should be extended to derive this number, but the absence of such data means that this currently stands as future work.

5.5.1 Implementation options

The calculation of the source counts represents a compare and add operation. This operation is a form of look-ahead computation since the result calculated in one router is only required in the next router. It therefore does not add to a router's critical path by requiring an extra pipeline stage, but can instead happen in parallel with other activities. In the currently modelled system, for a flit passing downstream, this value is calculated in parallel with that flit's crossbar traversal. A saturating adder computes this sum as the source-count values of itself and any other queued packets going to the same destination.

Several optimisations are possible to calculate this value. Firstly, the flow-based VC allocation mechanism ensures that only a single active packet from a destination flow exists at each input port, thereby limiting the number of elements that need to be supported by the adder. For a P-port, V-VC router, a module will be necessary for each input port, to compare an outgoing flow ID to that of all other active packets. A V:1 multiplexor can then be used to select the correct source-count. The found source-counts can then be summed with a P entry saturating adder for each output port. For a 5-port, 8-VC design with 6-bit flow identifiers and 3-bit source counts the two basic units of source-count selector and source-count adder (shown in Figure 5.9) were synthesised in a 90nm library. Both units resulted in delays of approximately 9.8 FO4. Even with additional wire delays, this can be accommodated within

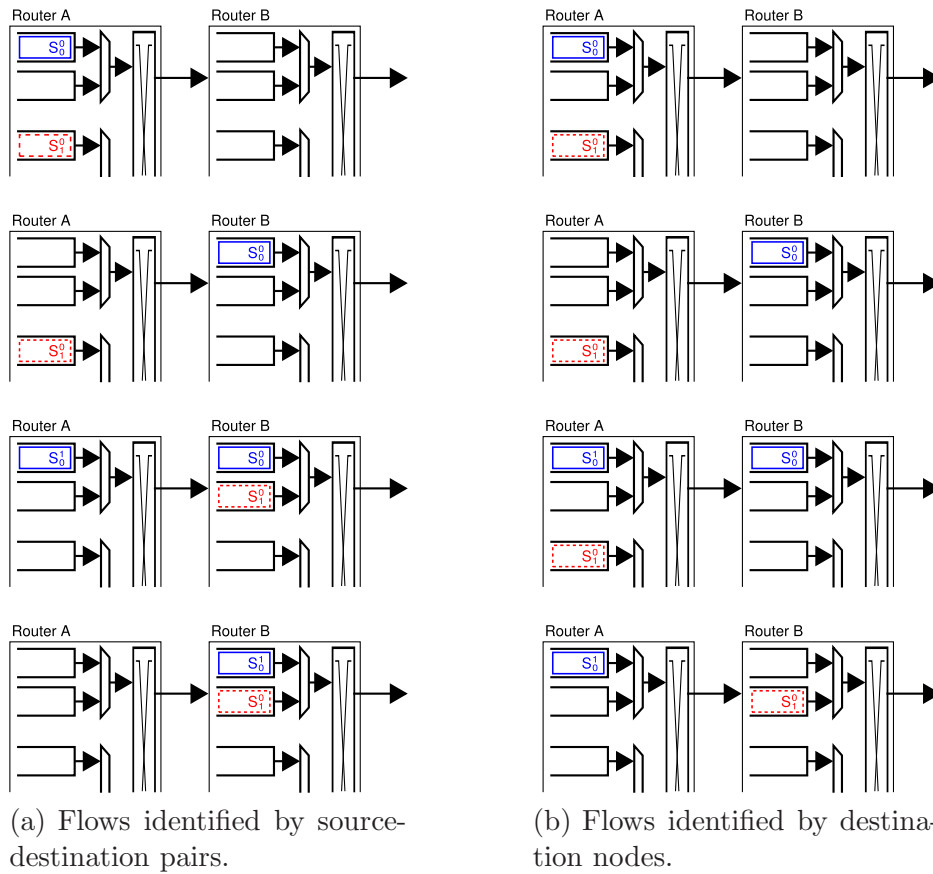


Figure 5.8: Different packet transmission behaviour with flows identified by source-destination pairs and those identified only by destinations, with S_y^x representing packet x from source y .

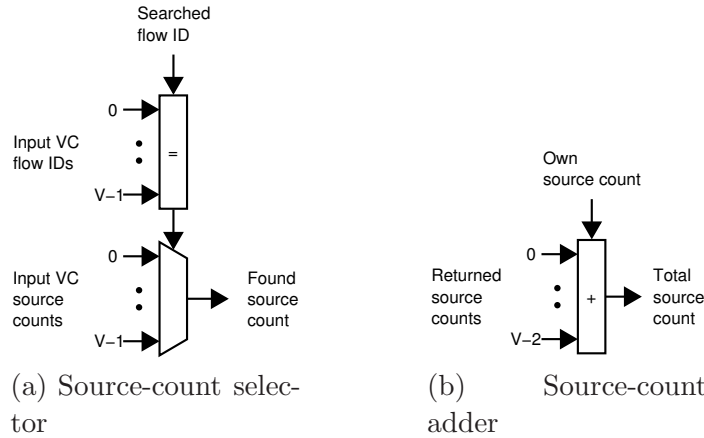


Figure 5.9: Basic units of source count selector and adder.

existing router clock periods of greater than 20 FO4.

5.6 Source-count based max-min arbiters

As described in Section 5.3, achieving max-min fairness with the approach demonstrated by Hahne involves building all allocators as units that serve active requests in a round-robin manner. In the case of a single arbiter with a number of requesters, a separate request/grant line is provided for each requester. The arbiter then effectively maintains a priority queue of the inputs with each input moved to the bottom of this queue once it is granted. Again, as already mentioned, in the case of NoC routers (based on matrix arbiters), this approach would identify flows as source-destination pairs, map them to VCs, and have the switch allocators provide a single request/grant line for each VC.

With the source-count modification, when a packet with a source-count greater than 0 arrives, it in fact represents several source-destination-flows, as described in Section 5.5. Assume that all of these source-destination-flows are merged into a single destination-flow which is then mapped to a single VC. If the switch allocator still provides a single request/grant line for each VC, only a single request/grant line is provided for multiple source-destination flows. If after every received grant the destination-flow were moved to the bottom of the arbiter's priority queue, source-destination flows would clearly not be served in a least recently used manner, as required to achieve max-min fairness. However, it is known from Section 5.5 that when a packet with a source-count greater than 0 arrives, subsequent packets to arrive for the same destination-flow will in fact be from different sources. A simple modification is therefore proposed to achieve the same behaviour as a single arbiter. Each destination-flow line in the arbiter is proposed to keep a *grant-count* field of the same width as the source-count fields (being 3-bits in this case). Every time a destination-flow is granted, its grant-count field can be incremented by 1. Instead of a destination flow line being moved to the bottom of the priority queue after every grant, this should now only be done when its grant-count becomes greater than or equal to the source-count of the granted input packet.

Consider an entire network controlled by such arbiters. In low traffic with all links underutilised, the work-conserving nature of the arbiters ensures that all flows receive their demanded rate. Consider now the case where a subset of all the flows in the network that pass through a single link increase their request rate such that that link becomes over-subscribed.

5. FAIR ALLOCATION TO FLOWS

Back-pressure will now be exerted on these flows so that they become back-logged, eventually limiting their injection rate at their sources. The VC allocation mechanism of the previous chapter also ensures that destination-flows progress through in a pipelined manner, within any buffering constraints. Upstream of this over-subscribed link, a steady-state therefore exists with the source-count fields remaining at fixed values. At the over-subscribed link, the source-count based arbiter described above will now iterate through packets of all source-destination flows, i.e. share the output bandwidth equally between them. In other words, this link will be the bottleneck link (with the max-min definition of the term bottleneck) for all flows passing through it.

If another source-destination flow were to arrive, demanding less than its fully fair share of the bandwidth, it will simply not be back-logged in the same way as the other flows. When a packet from this flow is present at the bottleneck link, it will eventually be served by the arbiter and the rest of the time the remaining bandwidth shared equally between the back-logged source-destination-flows.

With additional over-subscribed links, the exact same reasoning also applies to every other flow in the network. In other words, all flows going through over-subscribed links will have a bottleneck link, thereby satisfying the requirement for a fully max-min allocation.

As has already been mentioned, with localised traffic the number of VCs and width of the source-count fields stays relatively independent of network size. This ensures that this design is scalable. Again as already mentioned, the fairness mechanism designed here utilises only local information (i.e. routers make scheduling decisions based solely on information present within the router). No global knowledge of the network is required. This means that fairness will be achieved independent of the network topology used. This additionally means that fairness will be achieved with any routing strategy, as long as a flow uses the same route for its entire lifetime. With an implementation of this proposed scheme with on-chip arbiters, functionality will primarily be added to the arbiter's state-update-enable computation. Within a clock-cycle, this can occur in parallel with the grant generation and the design is therefore expected to have limited additional delay penalties. Moreover, the only modifications performed on the grant-counts are simple *increment by 1* or *reset to 0* operations. Finally, the impact of the potentially more complex comparison operations are also limited by the selection of narrow, typical-case counter widths. Although many designs trade-offs are clearly possible, the above reasons lead to the belief of this chosen method representing a good compromise design.

5.7 Modifying separable allocators

As described in Chapter 2, separable allocators represent a consensus design, providing low-delay allocation in NoCs. To extend fairness support to on-chip flows, this section therefore proposes modifications to separable allocators based on the arbitration strategies discussed above.

5.7.1 Modifying input arbiters

To achieve a fair allocation, the input arbiters must serve input source-destination-flows in a least-recently-used fashion. However, this is complicated by packets of the same destination-flow arriving on different VCs, as described in Chapter 4. If the arbiter were to operate in the *VC-plane* (i.e. each request/grant line of the arbiter were associated with a fixed VC), the priority-orderings maintained for the VCs would not be the same as that which needs to be maintained for flows. The arbiter therefore needs to operate in the *flow-plane* (i.e.

each request/grant line of the arbiter should be associated with a particular flow). Some mechanism is then required to map switch-requests of the input packets from the VC-plane to the flow-plane. Unfortunately, unlike the update calculation, this cannot occur in parallel to the grant calculation and may therefore represent the most significant delay penalty for this fairness scheme.

In the design developed, this is enabled by providing an *input-arbiter-table* at each input port. Each row of this table is associated with a particular request/grant line of the input arbiter and stores the flow ID currently active at that line. Active packets at the input need to search this table with their flow ID to find their associated flow-line. The observation that only a single flit can arrive at an input port per cycle allows the number of searches in this table to be minimised. In the current design, only a newly arriving head flit's flow ID is compared to the stored flow IDs. If a match occurs the incoming VC ID is stored in the matching flow's input-arbiter-table row. The optimistic nature of the flow-freed signal described in Section 4.3.4, implies that a second packet of a flow could arrive and means that each flow needs to be able to point to 2 input VCs. Using a FIFO structure for these two pointers will mean that the second packet will be blocked until the first clears, improving the resource utilisation efficiency. A delay analysis of the table search resulted in a relatively large delay of around 6.8 FO4. Techniques such as rapid transmission of these signals with the use of fast-wires may therefore become necessary to limit the delay penalties.

If a newly arriving flow is not found in the input-arbiter-table it signifies a change from a steady-state of a fixed number of flows going through the router. In this case any flow line not currently mapped to an active VC can be assigned to this new flow. Given a fixed number of active flows, the flow mappings will soon reach a steady-state once again. In the current system, the first inactive arbiter flow-line is linked to a VC when a packet arriving on it is not found in the table. The selection of this first empty flow does not significantly affect the delay as this can be computed early in the clock-cycle.

In the switch-allocation cycle the first operation required is to map the switch-requests from the input VCs to the relevant flow-plane line of the arbiter. A delay-analysis showed this mapping to be quite low-cost at only around 4 FO4 delay. Nonetheless, this may again be important as it cannot happen in parallel with the grant generation but instead needs to occur before it. After this request mapping, the grant generation occurs in the same fashion as in existing arbiters as described in Chapter 2. In parallel, the input VCs' source-counts are mapped to the flow-plane to participate in the key state-update-enable computation. This additionally requires the 3-bit grant-counts to be stored at each input-arbiter table row. The mapped source-counts are compared to these grant-counts with a greater-than-or-equal-to operation to result in a *flow-update-enable* signal being generated for each flow. Delay analysis of this comparison shows it to take only around 4.1 FO4 delay. As in existing arbiters, these are ANDed with the corresponding flow-plane grants to signal a flow-update event for that flow. Any successful flow updates combined with an active output stage grant and tail-sent condition gives the final update signal for the input arbiter.

A further complexity arises from the 2-stage structure of the allocators given the wormhole switching nature of the arbiters. Consider the flow pattern in Figure 5.10, where 3 source-destination flows, T , S_1 and S_2 , with S_1 and S_2 belonging to the same destination flow request at the shown input and output arbiters. At a time slot when the flow table (described in Chapter 4) allows S_1 to request for access to the switch it may not be granted by its input arbiter as it instead grants a packet from T being currently switched in a wormhole manner. A packet from S_2 will therefore be forwarded downstream. When the input arbiter finishes serving T , S_1 will now be blocked by the output arbiter serving S_2 in a wormhole manner. With high request rates from T and S_2 , this cycle can continue and S_1 will therefore get

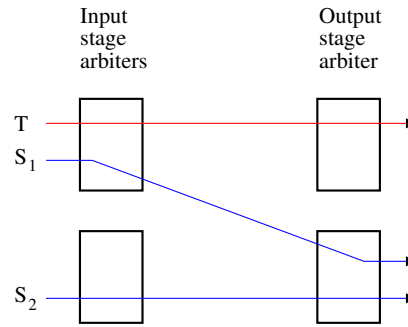


Figure 5.10: Traffic pattern to highlight potential starvation problem with flows.

starved out. The input arbiters must therefore not operate on a wormhole-switching basis. For non-flow-based traffic, this may result in higher delay at high traffic loads. However, the results of Section 5.8 show that the increased resource utilisation efficiency achieved by the design eliminate a large part of this potential overhead.

As already discussed, the grant-count values themselves are only ever reset to 0 or incremented by 1. A counter is reset to 0 if a new flow comes in to the router that is not found in the input-arbiter table, or the flow-update-enable signal is active for that flow and a flow-grant occurs. If a flow-grant occurs and the flow-update-enable is not active, the grant-count is incremented by 1. The main components of the input arbiter are shown in Figure 5.11.

5.7.2 Modifying output arbiters

The output arbiters need to serve all source-destination-flows across all input ports going to that output port in a least-recently-used fashion. Ideally, a single $(P \times V):1$ arbiter would exist at each output port. However, the high cost of such large arbiters makes this approach unfeasible. As described in Section 2.2.4 the output arbiters in separable allocators instead provide a single request/grant line for each input port, effectively grouping together all input VCs at an input port and maintaining an ordered queue of these input ports. This is clearly not the same as maintaining a priority-queue of all input flows and certain situations therefore exist where the proposed output arbiter allocations are not fully max-min but only an approximation of it. Potential solutions to this are further discussed in Section 5.9.

To implement the source-count based modification discussed in Section 5.6, an input-port should only be moved to the bottom of the arbiter's priority queue when the number of grants that have occurred to that input port exceeds the total source-count at that port. To avoid the need for large adders to calculate these total source and grant counts at every input port for every output port, a simple modification, based on the input arbiter table, has been developed.

A single 3-bit *output-grant-count* field is added to each row of the input-arbiter table described above. Source and grant count comparisons are then performed for individual flows and the results ANDed across all flows. Another single 3-bit comparator is therefore provided for each input flow which compares the source-count for that flow to the stored output-grant-count. If the output-grant-count is greater-than-or-equal-to the source-count a *flow-output-update-enable* signal is activated for that flow. Since the output port requested by a flow is also known, each flow can produce a *flow-output-N-update-enable* for each output port N . The relevant lines of these signals from every input flow can then be ANDed to

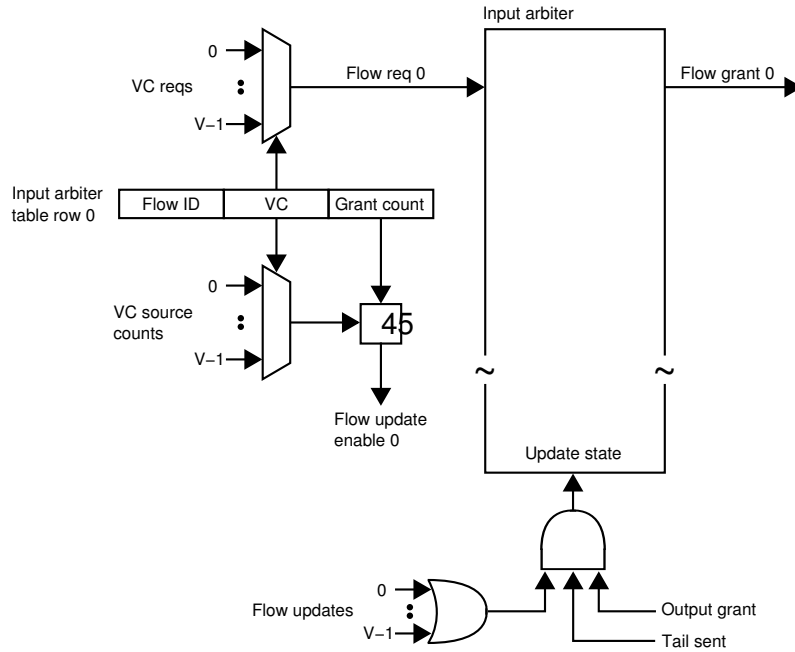


Figure 5.11: Modified input arbiter structure.

produce a *port-update-enable* signal for each output at that input. Since a flow must enable such a port-update for an output either when it is not requesting for an output or when it is requesting and it has an active flow-output-update-enable signal, the flow-output-N-update-enable is calculated as (flow-output-update-enable OR !output-port-N-required). The main features of this modified arbiter structure are shown in Figure 5.12. The delay of the 3-bit comparators is the same as described in Section 5.7.1 at around 4.1 FO4. Combining with the other simple AND, OR and INV operations will ensure that sufficient time is available for this update computation to occur in parallel with the main grant generation.

The update of the individual output-grant-counts is again restricted to either reset to 0 or increment by 1 operations. The count for a flow is reset to 0 either when a newly arriving packet is not found in the input-arbiter table and is mapped to that flow or when an output-update occurs at that input port for the relevant output port. If on the other hand an output update does not occur and the flow's flow-output-update-enable is not active, its grant-count is incremented if it receives a grant at both the input and output stages.

In the currently modelled system, 6-bit flow identifiers, 3-bit input and output grant counts each and 3-bit VC identifiers results in an additional 18-bits being required at each input VC for each input arbiter table row. Comparing this to the 2 nominal flit buffer locations for every VC in the base-case router with 128-bit flits represents a state-holding overhead of around 7%. The limited additional size of the flow-table at the outputs and the low observed areas for the synthesised portions of the design then further lead to the belief that the overall design can represent an area overhead of much less than 10% compared to the base-case router.

5.8 Results

This section evaluates the new router design (referred to from now on as the SDF router) with a number of traffic scenarios that result in unfair allocations in existing router designs.

5. FAIR ALLOCATION TO FLOWS

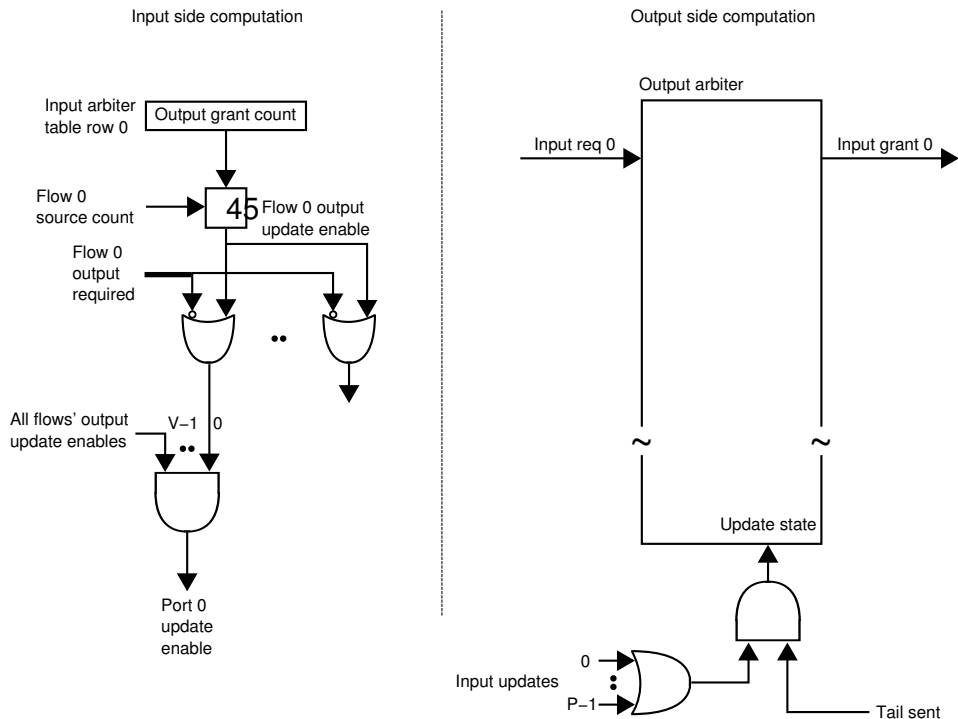


Figure 5.12: Modified output arbiter structure.

The same setup as before of 400 warm-up packets and 2000 measurement packets sent by each node is again used to obtain these results. As in Section 5.2 we start with the traffic pattern demonstrated in Figure 5.2. Table 5.3 shows the achieved allocations for the base-case, destination flows and source-destination flows routers with this traffic. As can be seen, only the SDF router divides the bandwidth equally between all the flows.

The importance of hot-spot traffic means that the network setup shown in Figure 5.4 is next used. Table 5.4 now shows how the SDF router again manages to equally divide resources between source-destination-flows. This test stresses the source-count based properties of the output arbiters, showing that they can indeed divide resources fairly.

The tests so far do not stress the source-count based fairness mechanisms in the input arbiters. To test these, the traffic pattern in Figure 5.13 was used, resulting in multiple destination-flows with different source-counts requesting at the input arbiters. Table 5.5 shows the rate allocations measured for the different router configurations. As can be seen, the input arbiters do indeed achieve a fair division of their resources.

All the tests so far have only involved fair divisions involving equal rate allocations to every flow. This does not test the demand of max-min fair allocation to allocate spare capacity to unfulfilled flows. To test for this, the same traffic pattern as in Figure 5.13 was again used, but this time source A only demands a rate of 0.1 flits/cycle and source B only demands 0.2 flits/cycle. The max-min allocation then meets both A and B's demands, with the rest of the bandwidth being equally divided to 0.23 flits/cycle between C, D and E. Table 5.6 shows that this is indeed achieved by the fair design.

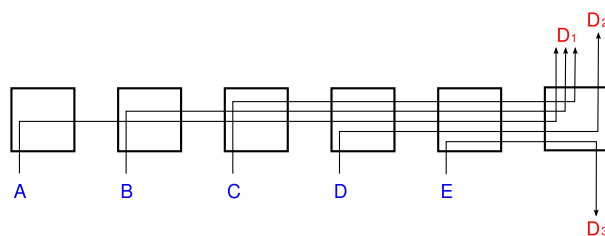
Finally, it is important that the new router does not deteriorate the performance of non-flow-based traffic. As in Section 4.3.5, the uniform-random traffic pattern was used to test for this. With an 8×8 network, the achieved packet latencies are shown in Figure 5.14.

Allocated rates (flits/cycle)			
Flow	Base-case	DF router	SDF router
A	0.02	0.07	0.14
B	0.03	0.07	0.14
C	0.04	0.07	0.14
D	0.08	0.07	0.14
E	0.17	0.07	0.14
F	0.33	0.33	0.14
G	0.33	0.33	0.14

Table 5.3: Fair allocations in SDF router compared to unfair flow allocations in base-case and DF routers.

Allocated rates (flits/cycle)			
Flow	Base-case	DF router	SDF router
A	0.06	0.06	0.20
B	0.06	0.06	0.20
C	0.13	0.13	0.20
D	0.25	0.25	0.20
E	0.50	0.50	0.20

Table 5.4: Fair allocations in SDF router for hot-spot traffic.



D_x Destination label

Figure 5.13: Network setup to test fairness mechanisms in input arbiters.

5. FAIR ALLOCATION TO FLOWS

Allocated rates (flits/cycle)			
Flow	Base-case	DF router	SDF router
A	0.06	0.06	0.20
B	0.06	0.06	0.20
C	0.13	0.13	0.20
D	0.25	0.25	0.20
E	0.50	0.50	0.20

Table 5.5: Rate allocations for test to stress input arbiters.

Allocated rates (flits/cycle)			
Flow	Base-case	DF router	SDF router
A	0.05	0.06	0.10
B	0.08	0.06	0.20
C	0.13	0.13	0.23
D	0.25	0.25	0.24
E	0.50	0.50	0.23

Table 5.6: Rate allocations for test with different flow request rates.

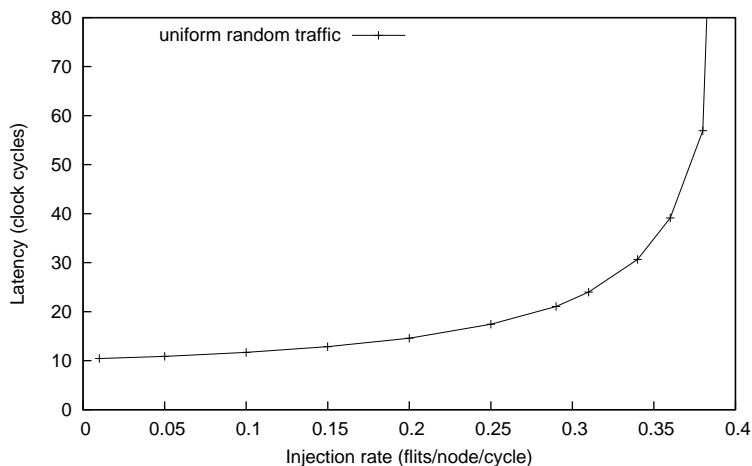


Figure 5.14: Uniform random traffic latency versus injection rate with SDF router.

Comparing these data to the results for the same experiment for the base-case router from Figure 4.4 and the DF router from Figure 4.18 shows that the SDF router closely matches the performance of those designs.

5.9 Future work

The fairness issues considered here raise a large number of significant new research questions for NoCs. The real-time nature of many of the important future classes of applications highlighted in Section 4.2 means that some form of Quality-of-Service support will be essential in future systems. As already discussed, it is a small step to change the relative importance of different flows in a fairness supporting mechanism to provide different service levels to them. Meeting such service demands within a single network with dynamic scheduling that also transports all bursty traffic can push up the total resource utilisation efficiency of the system. In the end this enables higher system performance to be achieved given a particular amount of a resource. Extending the existing system to allow such flow importance labelling with some form of associated admission control is therefore an important future research direction.

The focus of this chapter has been the provision of max-min fairness, given the lack of any more complex hardware-software interfaces at present. As discussed in Section 5.3, different definitions of fairness exist, with the α -fair framework providing a unified framework for all of them. Moreover, it is known that different values of α make different efficiency-fairness tradeoffs [56]. On-chip, which of these tradeoffs result in the highest overall performance is an open-question and needs to be further investigated. As Section 5.3 also discussed, many implementation choices exist to achieve fairness and a more detailed investigation of mapping these on-chip needs to be carried out.

Again operating within the limits of existing hardware-software interfaces, this work identified flows solely by physical node addresses. It is unproven however, that this represents the best cost-benefit trade-off point. A holistic research approach to identify the basic unit of computation and communication (such as a thread) and then provide fairness support at that level of granularity is therefore important.

More immediately, a thorough proof that the methods presented here result in max-min fairness is needed. The relatively high cost of the need to translate from the VC-plane to the

5. FAIR ALLOCATION TO FLOWS

flow-plane means that other, more static mechanisms to map flows to VCs can be of use. The many other mechanisms to achieve fairness on-chip also need to be considered further. The overall approach taken here of only modifying the base-case router to achieve the additional functionality leads to several problems, such as the non-exact max-min allocation of the output arbiters, etc. If instead, a much more basic starting point is considered, an entirely different design may result. Research into such simpler NoCs, such as those enabled by elastic interconnects [59], can be built upon to result in much simpler interconnection nodes that greatly simplify the arbitration problem at each node.

5.10 Summary

In any large shared resource system, a fair division of the resources across the different agents is essential. Given the importance of on-chip communication flows, this chapter has introduced the concept of fairly dividing NoC resources between flows.

The work first showed that the packet-based strategies of existing NoCs fail to correctly divide resources between flows. An important reason for this was shown to be the lack of any ability to identify flows. The VC allocation mechanisms of the previous chapter allowing flows to be identified by destination nodes was therefore upgraded to allow flows to be identified as source-destination pairs. All packets then carried an additional source-count field representing the number of sources transmitting to the same destination. This presented sufficient information to the allocators for them to identify source-destination flows.

Arbiters in existing NoCs were then extended such that they divided bandwidth across incoming flows in a max-min fashion. The design was then shown to achieve the fair rate allocations with only modest increases in clock period and an expected area overhead of less than 10%. A difference between the fair and base-case allocation rates of up to 233% in the demonstrated experiments highlights the potential benefits of achieving fairness. Moreover, in the absence of flow-based communications, the system closely matches the performance achieved by existing designs. This then achieves the original aim of the work of providing a single network, able to dynamically deal with a wide range of traffic patterns to ensure maximum resource utilisation efficiency, resulting in maximum system performance.

The fairness issues considered here raise many additional questions and present many alternative design solutions. A very large scope for future work therefore exists.

Conclusions

6.1 Thesis summary

The ever-shrinking distance reachable on-chip within the same number of gate-delays as geometric scaling continues is forcing a move towards more parallel and distributed computation architectures. Simultaneously, an increasingly limiting power-constraint is demanding higher power-efficiency from these architectures. Within this environment, some form of Network-on-Chip appears to be the only scalable way to provide communications on a chip. Separately, the increased computation power is also enabling a new set of application domains which share the properties of being highly parallel, more static and perform large data-transfers.

This thesis began by seeking a power-efficient design direction for NoCs. Such a direction was found by performing accurate power characterisation of a range of NoC architectures in Chapter 3. A static circuit-switched network, a wormhole network, a semi-dynamic virtual-channel network supporting QoS and a speculative virtual-channel network were synthesised, placed and routed in a CMOS 90nm, high performance technology. The extracted parasitics allowed accurate power results to be obtained. The results showed that NoC router power was a significant overhead beyond the link power and also appeared comparable to contemporary embedded processors. In the absence of other scalable communication infrastructures, these results fundamentally alter the cost of communication relative to computation on a chip. With long-haul, cross-chip interconnect dissipating a huge amount of power it may now be prudent to perform more computation to optimise global communications. A detailed analysis of the energy cost of routing a packet through a router next showed it to be dominated by the data-path and not the control-path in the network. This result extended the computation-to-communication energy ratio to the network routers, showing that the use of more complex control to optimise communication is justified from an energy perspective. A set of performance characterisations for the different designs were combined with the energy results to show that, from an energy-efficiency perspective, the benefits obtained from more complex control outweigh the increased power demands, further justifying their use.

Chapter 4 continued by investigating the communication patterns expected in NoCs in the future. It was shown that the kind of applications becoming more prominent lead to the presence of stream-like *communication flows* on-chip. Achieving high-performance for such applications requires NoCs to provide efficient resource utilisation in the presence of such flows. The packet-based allocation methodologies in current NoCs do not provide any flow-aware scheduling policies. It was demonstrated that ignoring flow-based information could result in highly inefficient network resource utilisation, given flow-based traffic. A mismatch in the transmission and reception rates between packets of the same flow means that network buffers and VCs are unnecessarily used up, blocking other flows from making progress. To solve this, flows were first identified as all packets going to the same destination. Moreover, with localised traffic, the number of flows expected to be seen at any node in the network in a typical case was shown to be limited. This enabled the development of a scalable allocation mechanism

6. CONCLUSIONS

where flows, identified by packets' destination addresses, were dynamically mapped to VCs in the network. In a typical-case, guaranteeing that at most one packet from a flow is active at a router input port at a time ensured that flows used the minimum required amount of resources. The achievement of this was confirmed for various flow-based traffic with the increased resource utilisation efficiency directly resulting in an associated speedup of up to 61.3%.

Chapter 5 addressed the problem of fairly dividing resources across flows in a NoC. The large volume of research in fair allocation within larger-scale telecommunication networks formed the basis of this work. As in Chapter 4, it was first shown that the packet-based fairness strategies of current NoCs do not result in fairness being achieved between flows. A recurring problem was shown to be the inability to identify flows. The mechanisms developed in Chapter 4 were then extended to enable router allocators to identify flows as source-destination pairs. For efficient buffer utilisation, flows were still identified and mapped to VCs according to their destination addresses, but packets carried a count of the number of sources attempting to transmit to the same destination in parallel. Existing arbiters in NoCs were then modified such that they served source-destination flows in a least recently used manner which resulted in max-min fairness being achieved for all supported flows in the network. In the traffic patterns tested, a difference of greater than 200% between the fair rate allocation and that achieved by existing designs for certain flows highlighted the criticality of achieving fair allocations.

The flow-aware scheduling mechanisms developed in Chapters 4 and 5 have been shown to provide large performance gains in the presence of flows. The power-efficient design direction derived from Chapter 3 achieved this with minimal expected additional power overheads. Additionally the design is expected to make only modest area and critical-path overheads. Beyond the clear benefits for flow-based traffic, the performance achieved for non-flow-based traffic is also close to being as good as existing designs. The original goal of this work of providing a single network that can efficiently schedule a variety of different traffic patterns over the same set of physical resources has therefore been achieved. The result of this thesis is a framework for all networks-on-chips to achieve previously unattainable performance levels with real applications.

6.2 Future directions

In the era of power-limited VLSI, maximum system performance can only be achieved through maximum power-efficiency. The extension of the NoC power-efficiency characterisation work begun in Chapter 3 therefore remains an important future research direction. Vast numbers of NoC designs have been proposed, but primarily only their performance characteristics have been evaluated. Instead, it would be much more meaningful to re-evaluate the performance returns of proposed NoC designs in terms of the power investment they demand.

With flow-aware scheduling in NoCs, how best to identify flows is a fundamental question that needs to be considered beyond the confines of existing hardware-software interfaces discussed in this work. Many levels of abstraction exist beyond the current hardware interface and it is not clear that ignoring this higher-level information represents the best power-performance trade-off. A detailed, joint hardware-software analysis of future systems and their communication patterns may identify the most beneficial definition of flows and the associated power costs of supporting them.

A similar detailed hardware-software co-evaluation could extend the fairness work begun in Chapter 5 to provide comprehensive Quality-of-Service support. The full spectrum of QoS requirements made by future application classes discussed in Chapter 4 first needs to be

investigated. Hardware fairness mechanisms along with the interface they provide to software entities can then be developed. This work would continue to mine the rich area of research of efficiently multiplexing all system communications on to a single set of physical resources, thereby achieving maximum system performance.

6. CONCLUSIONS

Bibliography

- [1] M. Ali, M. Welzl, and S. Hellebrand. A dynamic routing mechanism for network on chip. *NORCHIP Conference*, Nov. 2005.
- [2] M. Allalouf and Y. Shavitt. Centralized and distributed approximation algorithms for routing and weighted max-min fair bandwidth allocation. *High Performance Switching and Routing, Workshop on*, May 2005.
- [3] M. Amde, T. Felicijan, A. Efthymiou, D. Edwards, and L. Lavagno. Asynchronous on-chip networks. *Computers and Digital Techniques, IEE Proceedings*, Mar 2005.
- [4] ARM. ARM926EJ-S overview. <http://www.arm.com>, 2008.
- [5] J. Bainbridge and S. Furber. Chain: A delay-insensitive chip area interconnect. *Micro, IEEE*, Sep/Oct 2002.
- [6] A. Banerjee, R. Mullins, and S. Moore. A power and energy exploration of network-on-chip architectures. In *Proceedings of First International Symposium on Networks-on-Chip*, May 2007.
- [7] A. Banerjee, P. Wolkotte, R. Mullins, S. Moore, and G. Smit. An energy and performance exploration of network-on-chip architectures. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2008.
- [8] N. Banerjee, P. Vellanki, and K. Chatha. A power and performance model for network-on-chip architectures. In *Proceedings of the Conference on Design, Automation and Test in Europe*. IEEE Computer Society, Feb 2004.
- [9] R. Berezdivin, R. Breinig, and R. Topp. Next-generation wireless communications concepts and technologies. *Communications Magazine, IEEE*, Mar 2002.
- [10] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1987.
- [11] C. Bienia, S. Kumar, J. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, October 2008.
- [12] T. Bjerregaard and J. Sparso. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, Mar 2005.
- [13] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, Feb 2004.
- [14] J. Boudec. Rate adaptation, congestion control and fairness: A tutorial. Technical report, Ecole Polytechnique Fédérale de Lausanne, March 2008.

BIBLIOGRAPHY

- [15] K. Chandra. *Statistical Multiplexing*. Wiley Encyclopedia of Telecommunications, Jan 2003.
- [16] X. Chen and L. Peh. Leakage power modeling and optimization in interconnection networks. In *Proceedings of the International Symposium on Low Power Electronics and Design*. ACM Press, Aug 2003.
- [17] Y. Chen, J. Chhugani, P. Dubey, C. Hughes, D. Kim, S. Kumar, V. Lee, A. Nguyen, and M. Smelyanskiy. Convergence of recognition, mining, and synthesis workloads and its implications. *Proceedings of the IEEE*, May 2008.
- [18] M. Dall’Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini. Xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In *Proceedings of the 21st International Conference on Computer Design*. IEEE Computer Society, Oct 2003.
- [19] W. Dally. Virtual-channel flow control. *SIGARCH Computer Architecture News*, Jun 1990.
- [20] W. Dally, P. Carvey, and L. Dennison. The avici terabit switch/router. In *Proceedings of the Symposium on Hot Interconnects*, Aug 1998.
- [21] W. Dally and C. Seitz. The torus routing chip. *Distributed Computing*, Dec 1986.
- [22] W. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. *Design Automation Conference, Proceedings*, 2001.
- [23] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.
- [24] H. Darabi, S. Khorram, E. Chien, M. Pan, S. Wu, S. Moloudi, J. Leete, J. Rael, M. Syed, R. Lee, B. Ibrahim, M. Rofougaran, and A. Rofougaran. A 2.4 GHz CMOS transceiver for Bluetooth. *Radio Frequency Integrated Circuits (RFIC) Symposium, 2001. Digest of Papers. IEEE*, Dec 2001.
- [25] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Symposium proceedings on Communications architectures & protocols*. ACM, Sep 1989.
- [26] K. Diefendorff and P. Dubey. How multimedia workloads will change processor design. *Computer*, Sep 1997.
- [27] J. Dielissen, A. Rădulescu, and K. Goossens. Power measurements and analysis of a network on chip. Technical note, Philips Research, Apr 2005.
- [28] Z. Ding, R. Hoare, A. Jones, D. Li, S. Shao, S. Tung, J. Zheng, and R. Melhem. Switch design to enable predictive multiplexed switching in multiprocessor networks. *Parallel and Distributed Processing Symposium, Proceedings. 19th IEEE International*, Apr 2005.
- [29] P. Dubey. Recognition, mining and synthesis moves computers to the era of tera. *Technology@Intel Magazine*, Feb 2005.
- [30] M. Eldridge, H. Igehy, and P. Hanrahan. Pomegranate: a fully scalable graphics architecture. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., Jul 2000.

-
- [31] M. Elgamel and M. Bayoumi. Interconnect noise analysis and optimization in deep submicron technology. *Circuits and Systems Magazine, IEEE*, 2003.
- [32] M. Galles. The SGI SPIDER chip. In *Proceedings of Hot Interconnects Symposium IV*, Aug 1996.
- [33] F. Gilabert, S. Medardoni, D. Bertozzi, L. Benini, M. Gomez, P. Lopez, and J. Duato. Exploring high-dimensional topologies for NoC design through an integrated analysis and synthesis framework. *Networks-on-Chip, Second ACM/IEEE International Symposium on*, Apr 2008.
- [34] D. Greenfield, A. Banerjee, J. Lee, and S. Moore. Implications of Rent's rule for NoC design and its fault-tolerance. *Networks-on-Chip, First International Symposium on*, May 2007.
- [35] E. Hahne. Round-robin scheduling for max-min fairness in data networks. *Selected Areas in Communications, IEEE Journal on*, Sep 1991.
- [36] A. Hansson, K. Goossens, and A. Rdulescu. Avoiding message-dependent deadlock in network-based systems on chip. *VLSI Design*, 2007.
- [37] S. Heo and K. Asanovic. Replacing global wires with an on-chip network: a power analysis. *Low Power Electronics and Design, 2005. ISLPED '05. Proceedings of the 2005 International Symposium on*, Aug 2005.
- [38] R. Ho, K. Mai, and M. Horowitz. The future of wires. *Proceedings of the IEEE*, Apr 2001.
- [39] H. Peter Hofstee. Power efficient processor architecture and the Cell processor. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*. IEEE Computer Society, Feb 2005.
- [40] M. Horowitz. Scaling, power and the future of CMOS. *VLSI Design, Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*, Jan. 2007.
- [41] J. Hu and R. Marculescu. Dyad - smart routing for networks-on-chip. *Design Automation Conference, 2004. Proceedings. 41st*, Jun 2004.
- [42] Nanoscale Integration and Arizona State University Modeling (NIMO) Group. Berkeley Predictive Technology Model and BSIM4. <http://www-device.eecs.berkeley.edu/research>.
- [43] N. Jerger, P. Li-Shiuan, and M. Lipasti. Circuit-switched coherence. *Networks-on-Chip, Second ACM/IEEE International Symposium on*, Apr 2008.
- [44] N. Jouppi. The future evolution of high-performance microprocessors. In *Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, Nov 2005.
- [45] M. Karol, M. Hluchyj, and S. Morgan. Input versus output queueing on a space-division packet switch. *Communications, IEEE Transactions on*, Dec 1987.
- [46] N. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. PhD thesis, University of Twente, Jan 2007.

BIBLIOGRAPHY

- [47] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, Jan 1997.
- [48] J. Kim, W. Dally, and D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. *SIGARCH Computer Architecture News*, Aug 2007.
- [49] J. Kim and D. Lilja. Characterization of communication patterns in message-passing parallel scientific application programs. In *Proceedings of the Second International Workshop on Network-Based Parallel Computing*, Jan 1998.
- [50] A. Kumar, P. Kundu, A. Singh, L. Peh, and N. Jha. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. *Computer Design, 25th International Conference on*, Oct 2007.
- [51] A. Kumar, L. Peh, P. Kundu, and N. Jha. Express virtual channels: towards the ideal interconnection fabric. In *Proceedings of the 34th annual international symposium on Computer architecture*. ACM, Jun 2007.
- [52] J. Lee, M. Ng, and K. Asanovic. Globally-synchronized frames for guaranteed quality-of-service in on-chip networks. In *Proceedings of the 35th International Symposium on Computer Architecture*. IEEE Computer Society, Jun 2008.
- [53] K. Lee, S. Lee, and H. Yoo. Low-power network-on-chip for high-performance SoC design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, Feb 2006.
- [54] Magma. QuickCap datasheet. <http://www.magma-da.com>, 2007.
- [55] M. Martin, P. Harper, D. Sorin, M. Hill, and D. Wood. Using destination-set prediction to improve the latency/bandwidth tradeoff in shared-memory multiprocessors. *Computer Architecture, Proceedings. 30th Annual International Symposium on*, Jun 2003.
- [56] L. Massoulie and J. Roberts. Bandwidth sharing: objectives and algorithms. *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Mar 1999.
- [57] A. Maxiaguine, S. Kunzli, L. Thiele, and S. Chakraborty. Evaluating schedulers for multimedia processing on buffer-constrained SoC platforms. *Design & Test of Computers, IEEE*, Sep 2004.
- [58] Giovanni De Micheli and Luca Benini. *Networks on Chips: Technology and Tools (Systems on Silicon)*. Morgan Kaufmann Publishers Inc., 2006.
- [59] M. Mizuno, W. Dally, and H. Onishi. Elastic interconnects: repeater-inserted long wiring capable of compressing and decompressing data. *Solid-State Circuits Conference, Digest of Technical Papers. IEEE International*, Feb 2001.
- [60] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, Oct 2000.
- [61] T. Mudge. Power: A first class design constraint for future architecture and automation. In *High Performance Computing*, Dec 2000.
- [62] R. Mullins. Minimising dynamic power consumption in on-chip networks. In *Proceedings of the International Symposium on System-on-Chip*, Nov 2006.

- [63] R. Mullins, J. Lee, and S. Moore. Selecting a timing regime for on-chip networks. In *Proceedings of the 17th UK Asynchronous Forum*, Sep 2005.
- [64] R. Mullins and S. Moore. Demystifying data-driven and pausable clocking schemes. In *ASYNC '07: Proceedings of the 13th IEEE International Symposium on Asynchronous Circuits and Systems*, Mar 2007.
- [65] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st annual international symposium on Computer architecture*. IEEE Computer Society, Jun 2004.
- [66] R. Mullins, A. West, and S. Moore. The design and implementation of a low-latency on-chip network. In *Proceedings of the 2006 conference on Asia South Pacific design automation*. IEEE Press, Jan 2006.
- [67] S. Muthukrishnan. Data streams: algorithms and applications. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, Jan 2003.
- [68] C. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. Yousif, and C. Das. Vichar: A dynamic virtual channel regulator for network-on-chip routers. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, Dec 2006.
- [69] S. Ogg, E. Valli, B. Al-Hashimi, A. Yakovlev, C. D'Alessandro, and L. Benini. Serialized asynchronous links for NoC. In *DATE '08: Proceedings of the conference on Design, automation and test in Europe*, Mar 2008.
- [70] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, Jun 1993.
- [71] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Transactions on Networking*, Apr 1994.
- [72] D. Park, R. Das, C. Nicopoulos, J. Kim, N. Vijaykrishnan, R. Iyer, and C. Das. Design of a dynamic priority-based fast path architecture for on-chip interconnects. In *Proceedings of the 15th Annual IEEE Symposium on High-Performance Interconnects*. IEEE Computer Society, Aug 2007.
- [73] J. Park, B. O'Krafka, S. Vassiliadis, and J. Delgado-Frias. Design and evaluation of a DAMQ multiprocessor network with self-compacting buffers. *Supercomputing, Proceedings*, Nov 1994.
- [74] D. Patterson and J. Hennessy. *Computer Organization and Design*. Morgan Kaufmann Publishers Inc., 2004.
- [75] L. Peh and W. Dally. A delay model and speculative architecture for pipelined routers. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*. IEEE Computer Society, Jan 2001.
- [76] picoChip. The Picochip picoArray Architecture. http://www.picochip.com/products_and_technology/picoarray_architecture.

BIBLIOGRAPHY

- [77] M. Pirvu, L. Bhuyan, and N. Ni. The impact of link arbitration on switch performance. In *Proceedings of the 5th International Symposium on High Performance Computer Architecture*. IEEE Computer Society, Feb 1999.
- [78] E. Poh and H. Ewe. IPv6 packet classification based on flow label, source and destination addresses. *Information Technology and Applications, Third International Conference on*, July 2005.
- [79] A. Ravi, B. Carlton, Y. Palaskas, G. Banerjee, R. Bishop, M. Elmala, R. Nicholls, I. Rippke, H. Lakdawala, L. Franca-Neto, S. Taylor, and K. Soumyanath. A 1.4V, 2.4/5 GHz, 90nm CMOS system in a package transceiver for next generation WLAN. *VLSI Circuits, Digest of Technical Papers. Symposium on*, Jun 2005.
- [80] S. Rixner, W. Dally, U. Kapasi, B. Khailany, A. López-Lagunas, P. Mattson, and J. Owens. A bandwidth-efficient architecture for media processing. In *Proceedings of the 31st Annual ACM/IEEE International Symposium on Microarchitecture*. IEEE Computer Society Press, Nov 1998.
- [81] M. Rutten, J. van Eijndhoven, E. Pol Egbert, G. Jaspers, P. van der Wolf, O. Gangwal, and A. Timmer. Eclipse: heterogeneous multiprocessor architecture for flexible media processing. *Parallel and Distributed Processing Symposium., Proceedings International, Abstracts and CD-ROM*, Apr 2002.
- [82] S.G. Shakkottai, R. Srikant, N. Brownlee, A. Broido, and K. Claffy. *The RTT Distribution of TCP Flows on the Internet and Its Impact on TCP-based Flow Control*. PhD thesis, University of Illinois at Urbana-Champaign, 2003.
- [83] T. Shanley and D. Dzatko. *AGP System Architecture*. Addison-Wesley Longman Publishing Co., Inc., 1998.
- [84] S. Shenker. Fundamental design issues for the future internet. *Selected Areas in Communications, IEEE Journal on*, Sep 1995.
- [85] M. Snir and S. Otto. *MPI-The Complete Reference: The MPI Core*. MIT Press, 1998.
- [86] K. Srinivasan, K. Chatha, and G. Konjevod. An automated technique for topology and route generation of application specific on-chip interconnection networks. In *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*. IEEE Computer Society, Nov 2005.
- [87] M. Stensgaard and J. Sparso. Renoc: A network-on-chip architecture with reconfigurable topology. *Networks-on-Chip, Second ACM/IEEE International Symposium on*, Apr 2008.
- [88] R. Stephens. A survey of stream processing. *Journal Acta Informatica*, Jul 1997.
- [89] W. Sun and K. Shin. TCP performance under aggregate fair queueing. *Global Telecommunications Conference, IEEE*, Nov 2004.
- [90] Y. Tamir and G. Frazier. High-performance multiqueue buffers for VLSI communication switches. *Computer Architecture, Conference Proceedings. 15th Annual International Symposium on*, May 1988.

-
- [91] M. Taylor, J. Kim, J. Miller, D. Wentzloff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, L. Jae-Wook, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpfen, M. Frank, S. Amarasinghe, and A. Agarwal. The Raw microprocessor: a computational fabric for software circuits and general-purpose programs. *Micro, IEEE*, Mar 2002.
- [92] W. Thies, M. Karczmarek, and S. Amarasinghe. StreamIt: A language for streaming applications. In *Proceedings of the 11th International Conference on Compiler Construction*, Apr 2002.
- [93] Tilera. The Tile64 Processor. <http://www.tilera.com/products/processors.php>.
- [94] J. Vetter and F. Mueller. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. *Journal of Parallel and Distributed Computing*, Jan 2003.
- [95] I. Walter, I. Cidon, R. Ginosar, and A. Kolodny. Access regulation to hot-modules in wormhole NoCs. In *Proceedings of the First International Symposium on Networks-on-Chip*. IEEE Computer Society, May 2007.
- [96] H. Wang, X. Zhu, L. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *Proceedings of the 35th Annual International Symposium on Microarchitecture*. ACM/IEEE, Nov 2002.
- [97] N. Weste and D. Harris. *CMOS VLSI design: a circuits and systems perspective*. Addison-Wesley Longman Publishing Co., Inc., 3 edition, 2004.
- [98] P. Wolkotte, P. Holzspies, and G. Smit. Fast, accurate and detailed NoC simulations. *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, May 2007.
- [99] P. Wolkotte, G. Smit, N. Kavaldjiev, J. Becker, and J. Becker. Energy model of networks-on-chip and a bus. *System-on-Chip, 2005. Proceedings. 2005 International Symposium on*, Nov 2005.
- [100] P. Wolkotte, G. Smit, G. Rauwerda, and L. Smit. An energy-efficient reconfigurable circuit-switched network-on-chip. *Parallel and Distributed Processing Symposium, Proceedings. 19th IEEE International*, Apr 2005.
- [101] J. Xi and P. Zhong. A Transaction-Level NoC simulation platform with architecture-level dynamic and leakage energy models. In *GLSVLSI '06: Proceedings of the 16th ACM Great Lakes symposium on VLSI*. ACM Press, Apr 2006.
- [102] J. Xu, W. Wolf, J. Henkel, and S. Chakradhar. A design methodology for application-specific networks-on-chip. *Transactions on Embedded Computing Systems*, Dec 2006.
- [103] R. Zamani and A. Afsahi. Communication characteristics of message-passing scientific and engineering applications. In *Parallel and Distributed Computing Systems*, Nov 2005.