EVALUATING THE COST-EFFECTIVENESS OF DYNAMIC-BALANCED ADAPTIVE WORMHOLE ROUTERS

-- Algorithm, Simulation and ASIC Design

A Dissertation

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment
of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

(Jimmy) Jinming Ge

August, 2001

EVALUATING THE COST-EFFECTIVENESS OF

DYNAMICALLY BALANCED ADAPTIVE WORMHOLE ROUTERS

Jinming Ge

Approved:

Chairman of the Committee
Martin C. Herbordt, Associate Professor,
Electrical and Computer Engineering

Committee Members:

Olin Johnson, Professor,
Computer Science

Jaspal Subhlok, Associate Professor,
Computer Science

Pauline Markenscoff, Associate Professor,
Electrical and Computer Engineering

Richard Barton, Assistant Professor,
Electrical and Computer Engineering

E. J. Charlson, Associate Dean,
Cullen College of Engineering

Frank Claydon, Professor and Chairman,
Electrical & Computer Engineering

to my parents

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the help of many people. First, I would like to thank my committee for their helpful suggestions. Dr. Olin Johnson taught me the numerical analysis. Dr. Jaspal Subhlok taught me about computer networks and suggested several of the representative graphs to be pulled from the vast amount of simulation data. Dr. Pauline Markenscoff taught me computer architecture. Dr. Richard Barton made the Computer Engineering Seminar interesting and made many helpful suggestions. Most especially, I would like to thank my advisor, Dr. Martin Herbordt. Besides teaching me about computer architecture and writing, he motivated the research presented in this dissertation. Without his patient guidance, I could not have finished the research.

The discussion and cooperation with other faculties and CAAD lab-mates also sparked the research progress. I would like to thank Dr. Ramkrishna Prakash who not only taught me advanced digital design but also offered many comments on my research work and Dr. Lennart Johnsson who gave me some clues on how to evaluate the key aspects of performance. I would like also to thank my lab-mates: Jade Cravy who helped me with his plotting software and numerous comments, and Honghai Zhang and Calvin Lin who demonstrated useful toolkits. Harry Le and Kurt Olin, senior engineers at Compaq Computer Corporation and part-time graduate students of CAAD, also kindly offered suggestions on the router ASIC design.

All my family members, especially my parents-in-law and my friends in SWJTU (China) deserve very special thanks for their encouragement that helped me develop discipline and motivation.

The research described in this dissertation was supported in part by the National Science Foundation through CAREER award #9702483, the Texas Advanced Technology Program under grant 003652-0424 and a grant from Compaq Computer Corporation.

EVALUATING THE COST-EFFECTIVENESS OF DYNAMIC-BALANCED ADAPTIVE WORMHOLE ROUTERS

-- Algorithm, Simulation and ASIC Design

An Abstract

of a

Dissertation

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment
of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

(Jimmy) Jinming Ge

August, 2001

EVALUATING THE COST-EFFECTIVENESS OF DYNAMIC-BALANCED ADAPTIVE WORMHOLE ROUTERS

-- Algorithm, Simulation and ASIC Design

(Jimmy) Jinming Ge

University of Houston

ABSTRACT

The cost-effectiveness of wormhole torus-networks is systematically evaluated with emphasis on new dynamically-balanced wormhole routing algorithms. These algorithms use balanced assignments of virtual channels to distribute the workload evenly throughout the network, resulting in significant performance improvement. In particular we enhance previous algorithms by removing the non-packet-co-residence restriction on the escape channel. We demonstrate that the algorithms are deadlock-free.

A two part evaluation environment is developed consisting of a cycle-driven simulator for high-level measurements such as network capacity and an ASIC design package for low-level measurements such as operating frequency and chip area. This environment is used to compare our routing algorithms with their counterparts under various network constructions and workload parameters. Routers are modeled, verified, synthesized and optimized targeting a state-of-the-art ASIC technology. It is demonstrated that the complicated control of adaptive routing is not necessary on the critical path of the router if sophisticated techniques, such as parallel decoding with optimal path generation and a hierarchical skipping design of the round-robin arbiter, are applied.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 General Background

Multicomputer systems are currently accepted as the primary solution for the grand challenge problems in high performance computing as indicated by the ASCI [1] (Accelerated Strategic Computing Initiative) program. However, with the steady improvement of process technology, they are also likely to be used as the underlying architecture in application specific coprocessor cards for domains where high computing power rather than low memory latency is critical.

The interconnection network becomes more critical as the single processor gets faster. The network can be described by its topology, routing and flow control. Topologies in general can be divided into two classes, direct and indirect networks. In a direct network, each switch is connected to a node, while only a subset of switches is connected to the nodes in the indirect network. K-ary n-cube networks [2][3][7][40][46][52], as a representative of the direct network, and MIN (multistage interconnection network), are representative of the indirect network. Both are popular in multiprocessor systems [48][71][72]. Although there is little difference between these two kinds of interconnections in a unified view [26], the central controlled switch in the indirect one may form a bottleneck for its scalability [69]. We prefer the k-ary n-cube, especially configured as a 2D torus, in this dissertation for two reasons: The torus has a preferred symmetric topology and inherent lower latency than a mesh and the mesh is just a special case of a torus.

There are basically three major switching / routing classes used in communication systems: circuit, store and forward, and cut-through. Cut-through is preferred and widely used in recent multi-computers [73] due its inherent low latency. Cut-through switching with relaxed buffer requirement (known as ***buffered wormhole*** switching) is an even more popular scheme discussed in this research.

The primary issue of routing strategy is whether the routing is deterministic or adaptive [31][53]. Although the logic of deterministic routing is simpler and the cost is lower than that of an adaptive one,

adaptive routing can perform better depending on whether the adaptivity results in a balanced traffic distribution in the network. Another important issue is the arbitration policy. Not only should the policy guarantee necessary routing properties (e.g. freedom from live-lock and starvation), it should also lead to a cost-effective design. Adaptivity in both virtual channel and physical link selection will be explored and several arbitration policies will be studied in this dissertation.

Interconnection network performance is critical for all of the multicomputer systems described above, but cost is much more critical in the design of multicomputer cards/chips. Although there have been many research results on the performance evaluation of interconnection networks [26], few of them considered the system cost; even fewer did evaluations based on state-of-the-art ASIC (application specific integrated circuit) technology. As deep-micron technology proceeds with Moore's law, multicomputers on a chip (a type of System on a Chip or SoC) become viable. A low-cost, high-performance network will lead to the capability of integrating more nodes on a chip and so improve the SoC performance. A unique feature of this research is that it systematically evaluates various design tradeoffs, not only for performance, but also for cost.

1.2 The Issues to Be Investigated

This dissertation explores communication aspects of multi-computer systems, especially networks based on Torus implementations of buffered wormhole routing [31]. Our research concentrates on the domain of applications with data parallel characteristics that need either small amounts of memory or have alternative fast I/O sources, for example, streaming or on-chip sensors such as those used in computer vision and graphics. That is, the problem itself is computation limited rather than low-latency memory space limited.

The possible secondary domain is a multi-computer chip that is part of a larger multi-computer system. As deep-micron technology continues to improve dramatically – die size doubles every three years and line width halves every seven years, we can expect more nodes, PEs and routers/switches, to be built into a single chip. Figure 1.1 shows two possible multi-computer SoCs (System-on-Chip). To avoid the memory space bottleneck for some applications, the SoC can be used as the co-processor of a fully featured processing element.

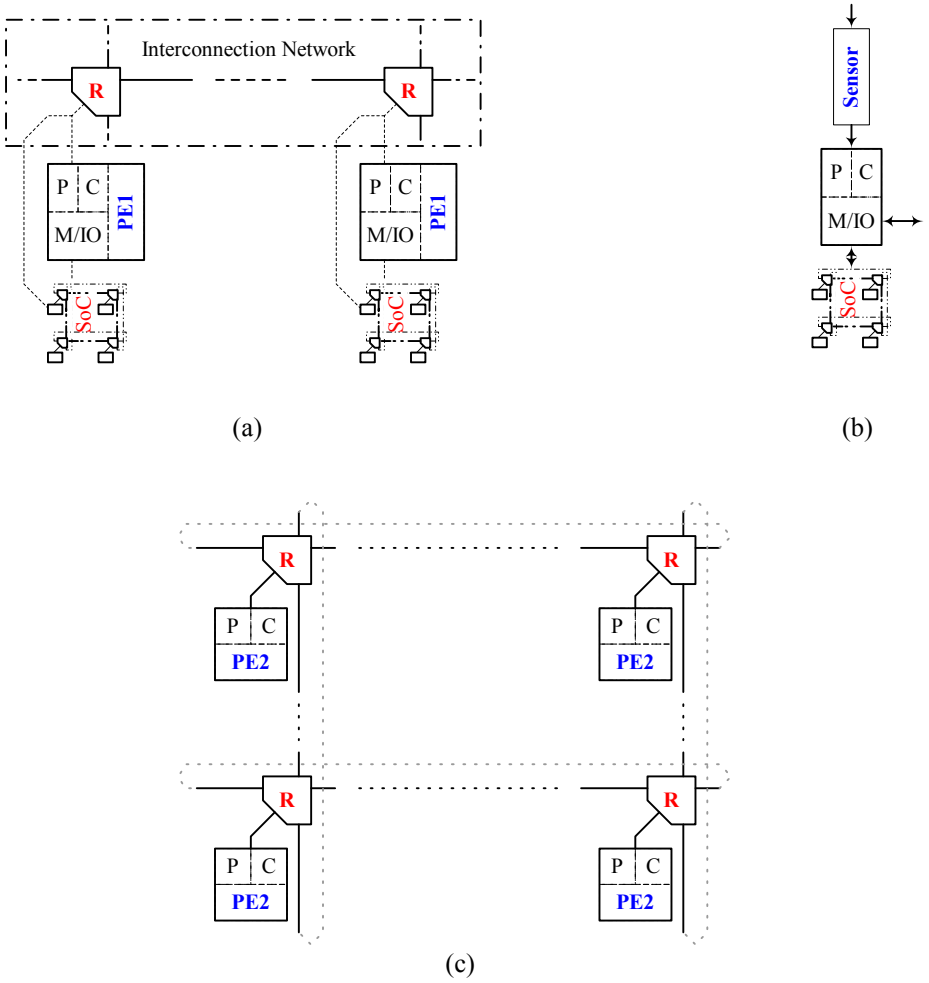


Figure 1.1. Shown is a plausible embedded SoC multi-computer system architecture. (a) SoC as a single node or co-processor in a general multi-computer system. (b) SoC as a co-processor in computer vision or graphics system. (c) SoC in a COMA architecture. R: router/switch; P: processor; C: cache; M/IO: memory and input/output; PE1: fully featured processing element; PE2: processing element without enormous memory and IO.

The cost of a switch/router in a SOC affects system performance more than in a general multi-computer system. It is obvious that a smaller network switch/router contributes to a smaller node; therefore

more nodes can be built into a SOC. On the other hand, if the SOC is fine-grained, higher performance is needed to tackle the relative higher communication overhead than in general coarse-grained system.

The basic issue of this research is to investigate the cost-effectiveness of various router designs. Besides the performance associated with each design, a key problem is how big each element should be. Following technology trends, we will present plausible scenarios for multi-computers-on-chip.

1.2.1 Routing Algorithms

Adaptive routing is preferred to oblivious routing under certain unbalanced workloads. The adaptivity can be defined in the scope of virtual channels, as well as partial or whole optional physical paths. The more adaptivity, the more improvement in performance, as long as it leads to a balanced distribution of traffic [25][80]. A key challenge of routing design is the deadlock problem. Deadlock arises when a circular dependency occurs, for example, when two packets are blocked and each is holding a resource required by the other. Deadlock can be prevented either by devising a deadlock-free algorithm [17][20][30][54] or released by a recovery scheme [11][12][26][61]. The most common and practical way to avoid deadlock is to design the deadlock-free algorithm because of the high cost and performance degradation of recovery schemes.

Several adaptive wormhole routing algorithms have been developed by using the concept of virtual channels (VC) and channel dependency graphs [20][21]. For example, Dally and Aoki introduced one of the fully adaptive algorithms by using k VCs for a k by k 2D-mesh [23]. The number of VCs doubles when used on a torus because of the wraparound connections. As presented by Aoyama and Chien, however, the virtual channels also substantially increase the router cost [5]. The most attractive algorithm of this type was devised by Duato [27] and a revised version has been integrated into the design of the Cray T3E [64] and the MIT Reliable Router [24]. This algorithm relaxed Dally's deadlock-free condition by allowing cycles to exist in the channel dependency graph as long as there is a routing sub-function whose extended channel dependency graph is acyclic. In this scheme, only three VCs per direction are necessary. Although it is designed to allocate the traffic evenly among virtual channels, a bottleneck link is formed. The 'non-co-residence' assumption on all buffers [26][28] also results in degraded performance for small packets.

Adaptive routing may not lead to better performance if the network resources, especially the virtual channels or buffers, are not used in a balanced way. This can cause even uniform loads to be unevenly distributed in the network [10]. Balanced virtual channel usage increases the probability that a packet can pass a blocked packet ahead of it, and also makes more efficient use of network buffer space. Scott [63] presented an off-line static balanced scheme for the T3D by introducing the concept of logical dateline. A packet can cross at most a single dateline. Routes not crossing any dateline are unconstrained and can use any virtual channel, while constrained routes can use only a particular virtual channel to cross the dateline. All of the unconstrained routes are assigned to the virtual channels to make the overall workload evenly distributed. The drawbacks of this strategy are as follows. First, it needs a lookup table to store the optimal virtual channel assignments, which is a burden for a large system – although the LUT can be implemented in the network interface, this just shifts the overhead to the packet formation phase. Second, its optimization is based on the uniform workload and is as a consequence less than optimal for most other workloads.

Buffered wormhole routing has gained popularity with improved VLSI technology. As more and more gates can be integrated in a single chip, it is possible to make the buffers larger and so release some congested physical links. A special case of buffered wormhole, the VCT (virtual cut-through) guarantees buffer space for incoming packets.

1.2.2 Network Design Space

A tremendous number of designs have been reported, but few are based on common network architectures and workloads [14][17][30][53]; it is therefore impractical to compare the performance reported. For example, to evaluate various algorithms, they should be based on similar network resources requirements (lanes, buffers and etc.). For fairness, we have developed a canonical router and network architecture that is suitable for all the cases of interest, and then compared the performance of each case using the same model under similar resource cost and workload environment.

Dozens of network and router parameters can be modeled in this canonical architecture. Here are some of the important tradeoffs discussed in this dissertation:

- The network size. There are two tightly coupled issues associated with this. One is whether the performance differences between different algorithms will scale up for a bigger sized network. If this is true, another issue is whether we can find a reasonable size to run simulations without changing the conclusion of which one is better.
- The number of lanes. It is stated [26] that more than four lanes would not be a cost-effective design, but the assumption is quite fussy. Given a certain amount of buffer space per node allocated into each lane, a more realistic concern is what the relation should be. As we define lane within the scope of VC for performance target, should the relation be explained as per virtual channel or physical link?
- The buffer space and its allocation. As we will explore multiple tradeoffs in the design, we can constrain one of them and try others – using the total node buffer size as a parameter rather than the size per lane buffer is a better choice. The effect of buffer size is mostly ignored in previous research [4][5][16][26][30]. Larger buffers enhance the performance as fewer links are held by a packet in WH routing, but too big a buffer can result in the HOL (head-of-line) problem and so restrict improvement. When comparing the WH and VCT as well as hybrid schemes, the key is to try out various possible ratios of buffer size over packet size to see the pros and cons of each in a more general scenario.
- The arbitration policies. This should be the core of flow control, but has rarely been touched in previous research. Some systems have been built with policies likely to result in starvation. For example, the PAR router [4] implemented with Highest-First scheme that tends to cause the packets in low-ordered channel to wait an unreasonably long time or never to proceed at all under certain types of traffic.

1.2.3 Workload Generation

Workload is generated in each local node and injected into the network to evaluate routing algorithms and network design tradeoffs. There are three issues that must be taken care of by the generator:

- Communication pattern. Besides the commonly used patterns such as uniform (random) and shuffles, we evaluate performance using more realistic patterns, such as dimension reversal and matrix transpose, as well as patterns reflecting differing degrees of locality, such as random near and local shift.
- Dynamic and static workload. To measure the throughput of a network, a dynamic load generator is commonly used by many researchers. Packets are generated with a specific communication pattern and mapped into the injector of each node, where they wait for available space to proceed. More of a concern with many practical applications, for example, a matrix transpose, is the time required for a communication phase. The load there is statically mapped onto each node before the operation.
- Packet size. It is almost as important as the communication pattern. When comparing buffered WH, VCT, or hybrid, it is actually the ratio of buffer size to packet size that is of the most concern. Here we assume the system has variable packet size and that VCT guarantees space for the largest packet.

1.2.4 Cost-effective Evaluations and the Tools

Results presented previously by researchers rarely take cost into account and even more rarely represent the cost quantitatively by implementation chip area and clock cycle time. Chien [16] proposed a cost-speed model based on the synthesis of block-oriented implementation of the PAR algorithm (on a mesh) with a .8 micron gate-array technology. It not possible to use the same process to evaluate the tradeoffs in our domain of interest. The reasons are as follows. First of all, different algorithms may require a different number of virtual channels and the cost is not only associated with VCs but also with the different implementations of control logic. And second, different topology and router architecture may have a significant affect on the cost and speed. Manufactures of certain machines [3][18][22][32][37][48][52] [63][64][68][74] may give some information about the cost (especially the speed) and performance (generally the bandwidth or throughput), but it helps little because of the different assumptions made on various machines. In other words, it is the lack of unified standards, general accepted software packages or tools that prevent direct comparison of cost-effectiveness of various tradeoffs.

1.3 The Objective and the Approach

The principal objective of this research is to evaluate the cost-effectiveness of various tradeoffs in the design of torus buffered wormhole routers. This involves several tasks. First, techniques are developed to enhance routing performance with dynamic balancing. Second, the performance of various design tradeoffs is evaluated on a common platform. Third, the implementation cost, chip area and clock timing are derived using standard ASIC design flow with respect to current technology.

To prove that the dynamic balanced adaptive routing algorithm is deadlock-free, we introduce the concept of the hardwired dateline, which is based on the analysis of previous algorithms, especially [28] and [63].

To evaluate the cost-effectiveness of various design tradeoffs in our design space, the performance is evaluated first by using a flit-level cycle-driven simulator developed for this research. The chip area and cycle time are reported after actually synthesizing the routers in a state-of-the-art ASIC technology.

The simulator is cycle-driven, as we believe it is more practical for our purpose. An event-driven simulator [47][62][81] is very slow due to enormous software overhead, although it may sometimes more accurately describe the behavior of a network. An analytical simulator [57] is ideal (both fast and accurate), but it is very hard, if not impossible, to model a large complicated network with enormous tradeoff options. Our simulator is modeled based on a canonical router architecture that is quite flexible for all kinds of algorithms, networks and workload parameters in the space this dissertation addresses. The performance is reported in flits per node per cycle for dynamic workload throughput and cycles for static workload finishing time.

The router design is modeled in Verilog HDL [58], and targeted to the LSI G11-p ASIC technology [49][50] with the logic synthesis package from Synopsys [75~77]. The dynamic-balanced fully adaptive and other wormhole routers are fully synthesized to obtain chip-area and clock cycle time. Finally, combining the simulation result and the synthesis report, we derive the cost-effectiveness of each router.

1.4 Contributions and Results

The cost-effectiveness of design tradeoffs is systematically obtained. Four of the major contributions are:

1. Evaluate design tradeoffs by considering cost and benefit, rather than just benefit.
2. Compare cost-benefit of adaptive routing with deterministic routing.
3. Present a dynamic balanced scheme and three routing algorithms derived there from:
    - Non-fully adaptive (DynBal) to contend with TRC, Duato's partial adaptive, and T3D-like;
    - Fully adaptive WH (F_DynBal) to contend with Duato's fully adaptive; and
    - Fully adaptive and hybrid WH and VCT (FD_DynBal) to contend with T3E-like algorithm.
4. Develop a common platform consisting of two parts: A cycle-driven simulator and a router ASIC design module.

The dynamic-balanced routing is shown to be deadlock-free by introducing the concepts of the hardwired-dateline and the escape channel. It outperforms the static-balanced scheme because of the flexibility of balancing under local traffic while static assignments can be either globally or hierarchically optimized but not both. It also outperforms the algorithms derived from cycle-tolerance theory by removing the restriction on escape channels.

The flit-level cycle-driven simulator is based on the canonical router architecture and has multiple options for network construction, resource allocation, arbitration, routing algorithm, and workload generation. The simulator is fast, flexible and accurate so that architects can explore the effects of various tradeoffs in multiprocessor design. It has a GUI interface and so is also convenient to use.

The router cost in chip-area and clock cycle timing presented here are obtained via actually implementing designs in a state-of-the-art ASIC technology. We compare the cost and performance of various options for the same platform and based on the same technology, rather than in different modeling regimes that use different assumptions, as was done in [16][30]. This establishes a fair environment for evaluation.

The other contributions are the results derived so far by using the cost-effectiveness evaluation platform. Their significance is that they can act as recommendations for the next generation of torus routing multiprocessors.

The major results of cost-effectiveness evaluation for buffered wormhole routers are as follows:

1. Constrained by low to medium chip area budget, under random workload, and independent of whether the traffic is uniform, local or non-uniform, the DynBal router is the most cost-effective and flexible one among the routers evaluated. Under the same cost, it
   - Outperforms TRC, Duato's PA by more than 30% on throughput (Duato's PA outperforms TRC only for low area budget).
   - Outperforms T3D-like by more than 10% on throughput, especially as packets get bigger or traffic more – even if the cost of the LUT is shifted out of router to source node. The difference of static workload delivery time between these two routers is apparently small, but if the off-router LUT in T3D-like routing is implemented by software, the extra LUT access time and packetizing delay will force T3D-like to take longer. DynBal is also more flexible than T3D-like because the latter has to be re-optimized for different network sizes and workloads.
   - Outperforms fully adaptive routers (either pure WH or hybrid of WH and VCT) if only low area budget is available. If medium chip area available, fully adaptive routing only performs slightly better than DynBal.
2. Constrained by a medium to high chip-area budget, under various workloads, F_DynBal is not only the most cost-effective but also the most flexible one. Under the same cost,
   - It outperform Duato's FA by more than 20% on throughput and 30% less latency on static workload delivery; this becomes more significant when the ratio of buffer size (cost) to packet size gets bigger.
   - It outperforms the hybrid fully adaptive routers, FD_DynBal and T3E-like under a medium area budget. Under non-random local workloads, it outperforms them by 25% and 40% respectively on throughput.
   - Although it may under-perform the two hybrid routers (FD_DynBal and T3E-like), the difference is small, especially when packets get bigger. It is more flexible than the hybrid routers.
3. Constrained by a high chip-area budget, FD_DynBal is preferable to T3E-like. Under the same cost (assuming the cost of LUT in T3E-like is shifted out of the router to local source node),

- Under uniform traffic, it has more than 30% lower latency for static workload delivery, although the throughput for dynamic workload almost the same as T3E-like. Under non-random local traffic, it also outperforms T3E-like by more than 15% in throughput.
- Although in some cases it uses more time to deliver the static workload than T3E-like, the difference is minimal if the extra time required by the latter for out-of-router LUT access and packetizing is considered.

Hence, we suggest that the DynBal be used as a low cost yet high performance router for random dominated traffic and that F_DynBal be used as a medium cost yet high performance router for broader band traffic (or synthetic workload).

These suggestions are derived after considering both the cost and the performance. For example, the performance simulation suggests a fully adaptive router for all kinds of workload because the high throughput measured in flits per cycle, while the cost evaluation suggests the TRC because of its low cost and high speed.

The other results are:

- Dynamic balanced-VC-assignment is more flexible than static balanced and enhances performance, especially for the case of low ratio of lane buffer size to packet size and for local traffic with lower cost.
- The power of each routing scheme depends on how effectively it uses the network resources. Considering performance only, the effect of lane number is related to algorithm used and so can be viewed in a new way:

  The performance of balanced routing with one lane per VC is better than that of TRC with two lanes per VC; the performance of fully adaptive routing with one lane per VC is better than that of partial adaptive routing or the VC balance within a single ring with two lanes per VC; however, doubling the number of lanes slows down clock rate by about 20% [36] and also results in a significant increase in chip area. Trying to compete with other algorithms by doubling the lanes is not likely to be cost-effective.

- Using both static workload and dynamic workloads is useful for performance evaluation. Monitoring the throughput variation when delivering static workload can help us to understand the inherent nature of routing algorithms, which is impossible with only a dynamic workload.
- Sophisticated design of control logic can remove it from the critical path. Adaptive routing is assumed to have low speed (clock frequency) and high cost due to its complicated control logic. Two methods presented in this dissertation, paralleling routing and path selection and hierarchic skipping logic in the design of the round-robin arbiter, significantly speed up the clock and lowering chip-area and so making the fully adaptive router most cost-effective.

Most of the discussion in this dissertation is based on the single-chip architecture, especially for the hardware cost evaluation. As the Fully Adaptive WH router with balanced scheme is small (2.60 mm square in LSI-G11p ASIC technology), it leaves much space for the processing elements so that more nodes can be built into a single chip. For example, a TinyRISC EZ4102 from LSI Logic Company takes 3.58 mm square. It has a compact 3-stage pipeline, compressed 16/32-bit architecture to reduce both code size and memory requirement, and a basic cache/memory/IO controller. A die of one cm square can integrate 16-node (router + TinyRISC PE). A reasonable die (15.8mm x 16.8mm) of this technology can hold 42 nodes. With 1MB on-chip SRAM available for cache memory, 23KB for each node, a COMA model may be enough for some applications. The VCT and hybrid WH schemes may require several thousand bytes of on-chip memory (rather than using registers) to hold packets in some/all virtual channels. In these the router will compete for use of the limited on-chip SRAM.

More advanced technologies will be available to integrate more nodes on a single-chip. Already the LSI-G12-p .18 micron technology has more than 4-times capacity of the G11 described above with 33 million versus 8.1 million (G11-p) usable gates. The PA-7300LC 32-bit RISC microprocessor has a die size of 4.69 square cm. Following the trends indicated by Moor's law, the capacity of a single chip will double in the near future; we can expect more nodes with more complicated PEs on a single chip.

1.5 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 presents an overview of related previous work on torus routing and evaluation; particular attention is paid to basic concepts. Dynamic

virtual channel balancing and two fully adaptive routing algorithms are discussed in Chapter 3. Chapter 4 presents the basic performance results of the algorithms developed as well other design tradeoffs. The chip area and speed as well as the cost-effectiveness of results are presented in Chapter 5. Finally, concluding remarks are given and further directions are discussed in chapter 6.

2 RELATED WORK

Three areas of previous work related to this research are presented in this chapter: router architectures, routing algorithms and evaluation schemes.

2.1 Router Architecture Tradeoffs

Although there are many architectural issues [59][67][79] that affect the performance and/or the cost of a design, we are concerned most about the issues that are tightly associated with the development of our canonical model. There are four issues that are critical to all the case studies of our interested domain: switching, buffering, header routing, and arbitration.

2.1.1 Switch Structure

The crossbar (Xbar) switch is widely used as the core of routers. It can be defined as a switching network with N inputs and M outputs, which allows up to min{N, M} one-to-one interconnections in the case of no contentions. Usually, N = M except for the case where inputs and outputs are different types of components (for example, processors and memory modules). A general logic representation of Xbar is shown in Figure 2.1.

The performance versus cost tradeoff leads to different Xbar implementations. Since single Xbars have a cost of O(NM), multiple Xbars must be integrated to implement large systems. In the IBM SP2 system [72], multiple 8x8-crossbar modules are used to construct s MIN (Multistage Interconnection Network). A few basic crossbars may also be used to design the router switch for direct-networks. For example, the PAR [17] uses two 4x4 switches in each adaptive plane and cascades the planar-switches to form the final switch when used for multiple dimensions. Cascading is also used in T3D [63] where inter-dimensional VCs are also used. It is worth pointing out that lowering the cost is often at the expense of performance. For example, cascading will require most packets to take extra cycles to get routed.

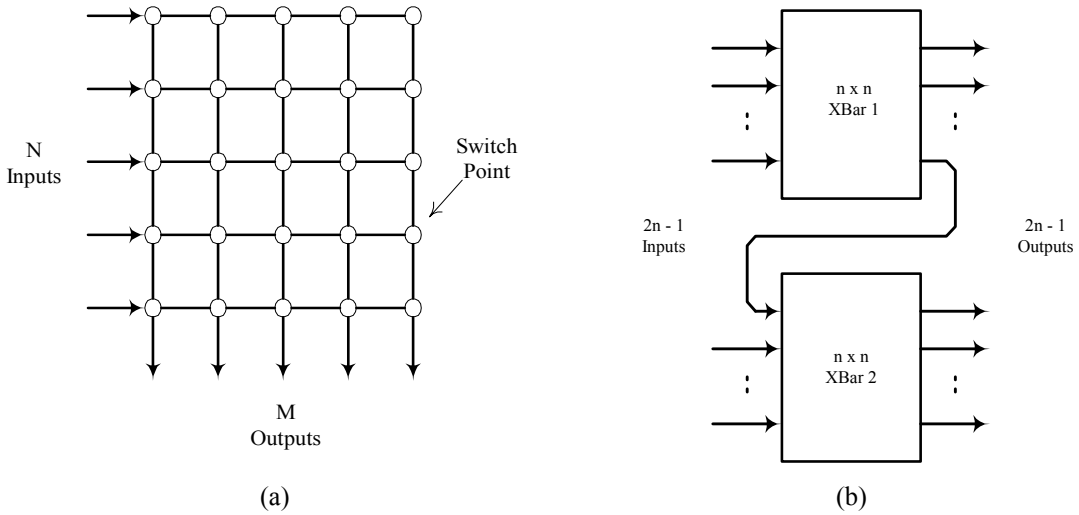


Figure 2.1. General crossbar switch architecture. (a). An N x M crossbar. (b). Cascaded switch

Cost-effectiveness can be better handled in systems with virtual channels, or lanes, than in the above general cases. The key here is that the lanes heading in the same direction generally share the same physical channel (PC). If each lane is corresponds physically to one of the inputs or outputs of a crossbar, as shown in Figure 2.2a, then the use of multiple lanes to enhance performance significantly increases the cost. If multiplexing (MUX) input lanes to one Xbar input and de-multiplexing (DMX) Xbar output into output lanes (used to de-couple Xbar allocation from PC allocation), as shown in Figure 2.2b, the Xbar complexity is greatly reduced. This will not degrade the performance because the average data rate out of the set of lanes associated with a given PC is limited by the bandwidth of this PC [23]. However, extra cost of control must be paid for the multiplexing. Figure 2.2c represents a moderate architecture that is constructed with lower complexity, but it is still questionable under some specific circumstances. First, if a standard (symmetric) Xbar were used in this approach, there would be several cross points wasted. Second, if the switch inputs were VCs rather than lanes, switching through Xbar and then de-multiplexing would not be a better choice than using a general multiplexer due to the complicated control introduced.

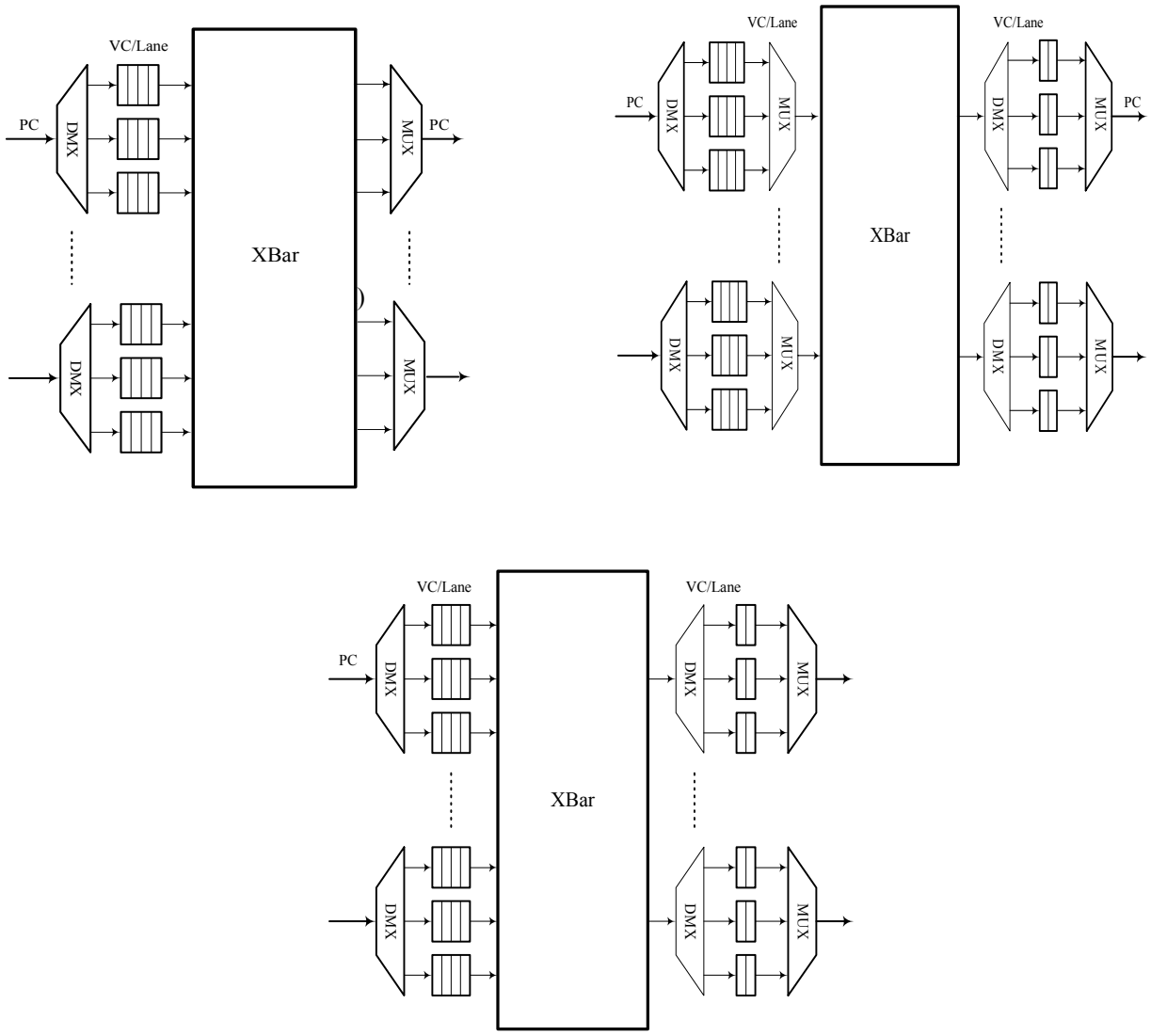


Figure 2.2. Crossbar switches used in systems with virtual channels or lanes. (a). A full crossbar. (b). Multiplexing / De-multiplexing I/O. (c). De-multiplexing outputs.

2.1.2 Buffering Scheme

There are two basic issues about buffering, the position and the size of the buffer.

Relative to the crossbar switch, putting the buffer before it (input buffering) suffers from the head-of-line (HOL) problem, in which a packet will block its followers even though they may head on different paths. More lanes can be introduced to alleviate the problem, but this solution has a cost in both the direct and indirect cost of switching. Buffering at the outputs (output buffering) is another solution to the HOL

problem [45]. Buffering at only one side, either input or output would force the control logic to arbitrate both the internal crossbar switching and the external physical link multiplexing. To speed up the switch cycle or pipeline different control stages, both the input and output buffers are required as shown in Figure 2.2b.

The other scheme developed to deal with the dilemma of whether to use input or output buffer is to make use of a central buffer that is shared by input and output as used by IBM [48][71][72]. If an incoming packet can proceed either because the central queue is empty or because it wins the priority over a packet already in the queue, it will be directly switched onto the output. Otherwise a packet in the central queue will hold the output link. More sophisticated management of the central queue such as Dynamically-Allocated Multi-Queue (DAMQ) [78] has been devised to enhance performance by allowing resource sharing among several lanes or queues.

The buffer size, a critical parameter for VCT because of the promise there to hold the entire packet, is also important for buffered wormhole routing. With the evolution of micron technology, more buffering space can be expected from a single chip. A big buffer can hold more flits of a packet so that a blocked packet ties up fewer links, and therefore benefits performance. For example, Cray T3E uses a much bigger buffer than Cray T3D [63][64]. As more chip area is used for buffering, however, less space is left for other performance enhancement schemes. For example, fewer nodes can be built in if a SOC (system-on-chip) is expected.

2.1.3 Header Routing

The basic issue here is whether pipelining is necessary and if it is necessary how deep it should be. The answer will actually depend on the system built and the technology used to build it.

An MPP system is generally fine-grained, which means that messages are generally sliced into medium-sized packets. The packet consists of a header flit, which holds the destination address and control information followed by data flits. When the header enters the switch, the destination information is decoded to select the path; the header may then be updated. Both the updated header and the data payload proceed via the path set up. In a non-pipelined router [4][16], each flit regardless of whether it is a header or not takes one cycle even though the actual time needed for these two different flits are quite different.

The resulting circuit is simple, but performance is lost. The loss of performance is much more noticeable as the packet gets bigger, and is much more concerned with the dramatic improvement of contemporary micron technology that inherently tends to ever increasing speed. So pipelining is much required for good system performance.

It is a natural choice to parallelize the path setup (routing and path selection) with the transmission of both header and data, although detailed implementations may be quite different. The architectural necessity of doing so is the buffering at both input and output. A depth-two pipelined model adapted from the original non-pipeline model [16] is used in [30]. Further deep pipelining is possible but we must cautiously consider the performance gain and the cost. The pipeline of routing control should be of such a depth that matches its cycle time with the flit transmission time.

2.1.4 Arbitration Policy

When multiple sources request the same target, such as a cross-point of the central crossbar or a multiplexed physical link, a policy must be applied to arbitrate the conflicts. The fundamental requirement of arbitration is fairness with low cost.

The simplest and therefore the lowest cost policy is arbitration by fixed priority. First-come-first-served (FCFS) is often used if the requests are queued. A variant of the policy is straight-first (SF) [4][16]; this best characterizes deterministic routing over k-ary-n-cube because a packet turns more rarely than proceeding forward. The weakness of these polices is that they can lead to starvation and thus degrade system performance.

More sophisticated arbitration policies can be devised that are starvation-free but have higher cost. Besides commonly used policies such as random and round-robin, schemes devised for general network switches can also be used for multicomputer network switches. For example, credit-based flow control [42] and iSLIP [51] are designed to enhance the performance of ATM switches. The challenge here is to get the simplest design while maintaining high performance.

2.2 Routing Algorithms

A fundamental requirement of routing algorithms is to be deadlock-free. If a packet already holds a resource and is still waiting for another packet to release its resource, and if cyclic waiting occurs, then no packets can proceed. There are two kinds of cycles in a torus network. One is formed physically by wrap-around links; the other is logically formed by the turns across dimensions. We begin with a discussion of standard solutions for these two cycles, present problems with those solutions, and then introduce schemes developed to solve these problems.

2.2.1 The Torus Routing Chip (TRC) and Its Algorithm

The first kind of cycle is the ring, which has a preferred symmetric topology -- each node can reach any node in the same ring. But the potential deadlock configuration, shown in Figure 2.3a, does not have an easy handling strategy.

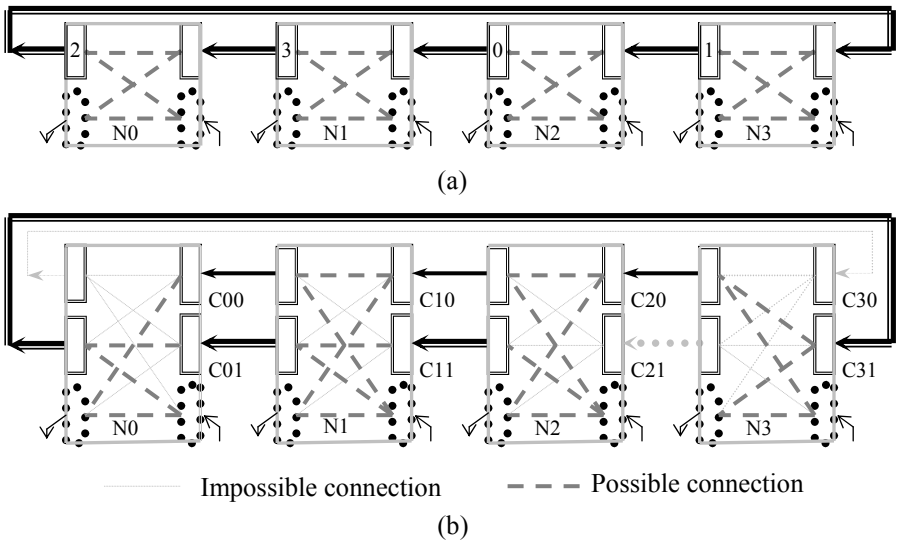


Figure 2.3. The deadlock and its prevention when routing in a single ring. (a). Deadlock configuration in a single ring. (b). Dally's deadlock-free algorithm for a single ring

Dally presented his deadlock-free algorithm in [20][21]; it introduces virtual channels and ***Channel Dependency Analysis Theory***:

A routing function, R for an interconnection network I, is deadlock-free iff there are no cycles in the channel dependency graph, D.

Each physical channel involved in a cycle is split into two virtual channels, or in other words, into two channels multiplexed into one physical channel, as shown in Figure 2.3b. The routing algorithm forces the virtual channels to be used in a fixed order. The packet proceeding to its destination (in the dimension) without wraparound uses virtual channel 0, otherwise it uses virtual channel 1. There are no cycles in the channel dependency graph, so deadlock is prevented.

The packet requiring a different virtual channel from that required by a blocked packet can bypass the blocked packet so that the virtual channel may also potentially contribute to the improvement of overall performance. However, this actually depends on whether the traffic is evenly distributed among the resources. It was found by Bolding [10] that the above algorithm distorts even a uniform workload because of uneven usage of the virtual channels.

The torus routing chip [21] was prototyped in 1985 using 3um CMOS technology with size 4.5mm*6.5mm, running at about 4MHZ. It was used for two-dimensional and unidirectional tori with a total of 5 lanes (Virtual Channels), 2 virtual channels (lanes) per dimension/direction. It had byte wide self-timed physical channels, a 2-cycle external and 4-cycle internal signaling convention, and achieved 64 Mbits/s for each dimension.

The TRC algorithm can be easily adapted for use in a bi-directional torus if the dependency between two directions is isolated first. To split the tie between the two directional physical rings of each dimension, minimal source routing must be used first for a packet in its source injector lane. Further, a packet must request the virtual channel in one of the directions to complete its route in a specific dimension (for DOR, X first), depending on in which direction the path is shorter. If they are of equal distance, the tie can be broken by even or odd numbering of the node in this dimension, which contributes to the balanced use of the VCs.

2.2.2 The Turn Model

Deadlock can also occur if a packet routes along a ring in a torus and makes turns to route in another ring belonging to a different dimension; four turns can form a cycle, e.g. East-North-West-South. Preventing enough and proper turns in routing can prevent deadlock from occurring. Dimension-Order Routing (DOR) is the simplest scheme in any oblivious routing, while other routing algorithms with

different adaptivity can be devised by the analysis of the turn model [34]. In the TRC algorithm, DOR is used across multiple dimensions.

2.2.3 Statically Balanced Virtual Channel Assignment

The virtual channel balance, or the relative traffic carried by each virtual channel, has a significant effect on network performance [10][80]. The balance increases the probability that a packet can bypass a blocked packet ahead of it and so more efficiently uses network buffer space.

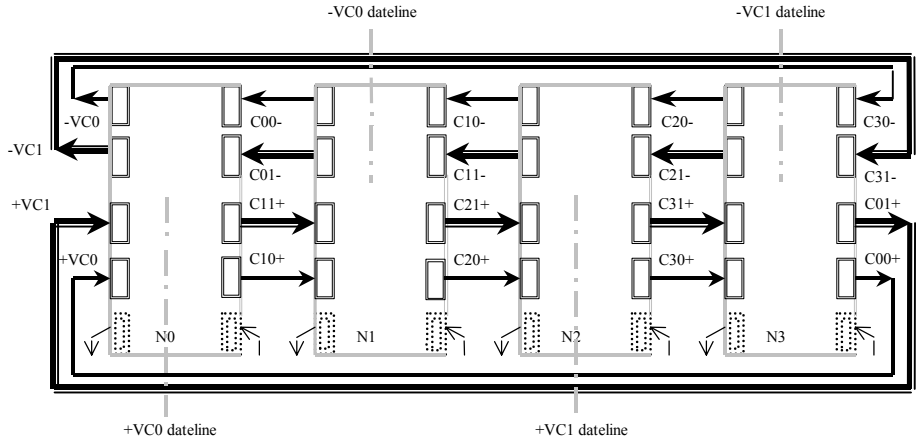


Figure 2.4. The concept of dateline used in T3D.

Scott [63] presented an off-line static balanced scheme for the bi-directional torus used by the Cray T3D. He introduced the concept of logical dateline, as illustrated in Figure 2.4. Each ring is considered independently and has an imaginary dateline crossing a node on each virtual channel along each direction. Workload is assumed to be uniform – each node sends packets to any node in the ring with equal probability. The route is fully determined by the source and destination address of a packet. A packet can cross at most a single dateline. Any route not crossing a dateline is unconstrained and can use any virtual channels, while all constrained routes must use only the particular virtual channel assigned to the dateline it is crossing. All the unconstrained routes are assigned the virtual channel by optimization distributing the overall workload evenly among the VCs on the ring. Instead of routing dynamically, the route is decided in the source node and the packet keeps using the same virtual channel until it reaches its destination.

The disadvantages of this strategy are as follows. First, it either needs a lookup table to store the pre-decided route or must embed the VC assignment in the packet header, which would be a burden for a large system and not be an option for a substrate chip. Second, its optimization is based on a uniform workload, meaning that it may behave suboptimally otherwise. Third, when the network needs to be reconfigured for a larger system, the assignments must be re-optimized and the lookup table refreshed. Lastly, the VC assignment is impossible to balance when the network width gets bigger: the VC assignment cannot be perfectly balanced (100%) if the width is greater than 4.

A router and multiprocessor system (T3D) [63] with 128-nodes arranged in an 8x4x4 torus was built in the 90s. The basic features of the router are 16 bit physical link data width (phits); 64 bit words; 3~26 phits per packet; and a 1 flit VC buffer. The system clock and network transmission rate is 150 MHz and the bandwidth per node 150MB/s (or 75MB/s per PE as there are 2 PEs per node). The routing algorithm is DOR. The commodity microprocessors used as PEs are the DEC Alpha 21064.

Both architecture and algorithm changes may be necessary for general use. The inter-dimension channels in T3D are used to cascade small switches, but it would not be necessary to use them if inter-dimension channel selection is done dynamically and based on availability. Other strategies can also be used to optimize the virtual channel balance as long as they follow the dateline rules.

2.2.4 The Cycle-Tolerance and Partially Adaptive Routing

Each of the three schemes analyzed so far, the TRC, the Turn Model, and the T3D, are based on the acyclic channel dependency to guarantee deadlock-free. The deadlock-free condition of the channel dependency analysis theory can be relaxed by allowing cycles to exist in the channel dependency graph as long as there is a routing sub-function whose extended channel dependency graph is acyclic [27]. For a packet that resides in a cycle, an escape path to an acyclic channel always then exists, so that it can be drained off in a certain number of cycles. To make this work, each virtual channel cannot co-reside more than one packet. In addition, the route (or channel) selection function and its underlying controller must be smart enough to check for temporary blocking and switch the packet to a free channel. For the convenience of further analysis, the *Cycle-Tolerance (Extended Channel Dependency Analysis) Theory* can be cited as the following [28]:

A connected and adaptive routing function R for an interconnection network I is deadlock-free if there exists a routing sub-function R1 that is connected and has no cycles in its extended channel dependency graph D_E.

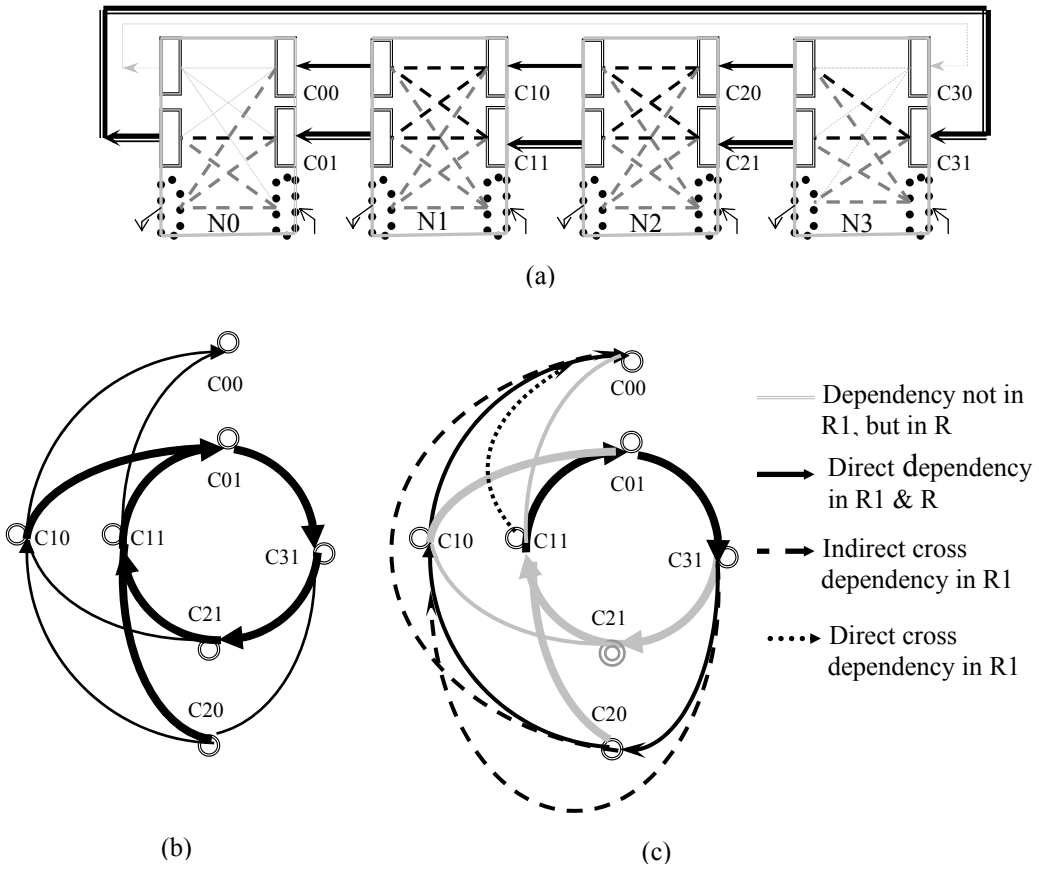


Figure 2.5. Cycle-tolerant partial adaptive routing. (a). The partial adaptive algorithm R used for IO buffer buffering. (b). The channel dependency graph of R. (c). The extended channel dependency graph of R1.

A partial adaptive routing algorithm described in [29], here called ***Duato's PA*** or ***ParAdapt***, can be derived from this theory. It uses TRC as a sub-routing function and adds more options for the channel selection function. Figure 2.5 illustrates the routing inside a single uni-directional ring. Any packets destined for a higher numbered node have to use the virtual channel Cx1 while others can use either Cx1 or Cx0. The virtual channel C30 cannot be used at all; this forms a bottleneck link from N0 to N3. This is one of the two drawbacks of this scheme.

The other drawback comes from one of the assumptions applied to all channels. As stated in the assumptions of this theory [26][28], no packet co-residence is allowed for any of the virtual channels. This means a larger buffer, which may be even bigger than a packet; more bubbles will be introduced; and, to put it another way, the extra buffer size will not be credited for performance. To fairly compare different schemes, we must assume the same buffer size for each. In this case, we give the extra buffer space (in excess of packet size) to the lane interfaced with the local injector and/or the ejector (actually they are not the virtual channel used for deadlock prevention) so the performance losses can be somewhat compensated.

2.2.5 The Fully Adaptive Routing

Although several fully adaptive torus wormhole routing algorithms [14][17][19][23][31] [66][80][82] have been presented, they require either an excessive number of VCs or complicated control. For example, for an n-dimensional torus, Jessope's scheme requires $2^{(n+1)}$ VCs per physical channel while Linder's scheme requires $(n+1)2^{(n-1)}$ VCs. As discussed below, the number of VCs significantly slows down the router and increases the cost in chip area [5].

Based on the cycle-tolerance theory, a practical fully adaptive routing algorithm [29], noted as ***Duato's FA*** *(*or ***F_ParAdapt)*** here, was devised with only one extra VC per direction added. This VC is used to cross dimensions in any order following a minimal path. The added VC does introduce a cyclic channel dependency, but as long as the extended channel dependency graph is acyclic, the fully adaptive algorithm is deadlock-free.

The assumption made for this algorithm is the same as for the previous one – a queue cannot contain flits that belong to different packets; the bubbles therefore still exist and worsen when there is a relatively large ratio of queue size to packet size. The bottleneck formed in the partial adaptive algorithm exists also, but may not be as severe as in the partial one because of the extra VC.

2.2.6 Hybrid Fully Adaptive Routing

In a general network (LAN or WAN), packet switching uses the store-and-forward (SAF) scheme. Each packet is completely buffered at each intermediate node and the header is extracted to determine the next hop. As it exhibits extremely high latency, SAF is not used in contemporary multi-computer networks.

What can be done is that the header and following data can be forwarded in pipeline fashion, or cut-through to the next hop, before the entire packet has been received as long as the link is free. Virtual cut-through (VCT) requires sufficient space to hold the whole packet in the case of congestion; however, its latency is generally much lower than SAF [26]. But the requirement that all channels can buffer complete packets makes it difficult to construct a small, compact and fast router.

Combining VCT with WH is implemented in the Cray T3E machine. The T3E routing is an adaptation of cycle-tolerant fully adaptive routing [64]. To use the T3D algorithm as a routing sub-function the dependencies between the two non-adaptive VC via the intermediate, adaptive VC must be removed so that the VC assignments in non-adaptive virtual network (VN) can be statically optimized. A scheme to do so is to replace the routing from the non-adaptive VN to the adaptive VC with VCT. The buffer in the adaptive VN is large enough (over two times maximum packet size) so that multiple packets (or flits belonging to different packets) can co-reside in it. A packet can be drained off from the non-adaptive VN entirely to the adaptive VN as long as enough space is available.

Besides the problems inherited from T3D, the T3E scheme does not permit arbitrarily large packets, and is therefore less flexible. Whether the combined scheme works better than a pure fully adaptive WH routing is still questionable.

Multiprocessor systems (T3E) [64] with a maximum of 1024-nodes arranged in an 8x16x8 torus were built in 90's. The basic features of the system are 14 bits physical link data width (phits), 5 phit flits, 64 bit word, and 5~37 phits per packet. The adaptive buffer size is 22 flits, while the non-adaptive buffer size is 12 flits. LSI Logic's 500K ASIC technology (.8um) was used to achieve a 75 MHz clock rate, a 375 MHz network transmission rate with 5X time-multiplexed transmission, and a 500MB/s bandwidth per node. Direction-Order routing is used in the non-adaptive network so that it offers a higher degree of fault-tolerance. Five rather than three VCs were used for each physical channel because of the separated request-response cycles.

2.3 Evaluations

Performance evaluation is generally done via cycle-driven flit-level simulations, with the results described in a BNF (Burton Normal Form) that best reflects the metrics [26][40][55][60]. To approximate

the design tradeoffs more reasonably in the real world, a few researchers have tried to model the delay of components of the router, and presented more meaningful cycle time for a small set of parameters given [16][30].

2.3.1. Metrics

The simulated performance is often described in the BNF that best reflects the well-known metrics. The X-axis of the BNF represents the normalized accepted traffic while Y represents the latency.

The accepted traffic is the amount of information delivered per time unit, which is often normalized by dividing network size and message length. In a cycle-driven flit level simulator, the traffic is measured in flits per node per cycle. Since different designs can result in different cycle times, we must be very cautious when using this measurement. To be close to the reality with respect to cycle time, the component delay can be modeled [16] and then the throughput (or traffic) can be measured in flits (or bits) per node per nanosecond.

The latency is the time elapsed since the message transmission was initiated until the message is received at a destination node. There are no standards to define when the transmission started and terminated [26], for example, whether the queue delay in the injector and ejector should be considered. The latency is measured in cycles in a cycle-driven simulator, although it can be represented in nanoseconds if the hardware is modeled. Often, it is the average latency that we are concerned with, rather than single packet latency.

2.3.2 Simulator

There are principally three types of simulators often used in the performance evaluation of interconnection network: cycle-driven, event-driven and analytical. Predominantly, performance evaluation is done via cycle-driven flit-level simulations. Simulation at the gate level or transistor level is very complicated, and the speed would be very slow, therefore considerably reducing the design space to be explored. The lack of hardware details in flit-level can be somewhat compensated for by the cost-speed analysis after the fast simulation having concentrated the design space. The most representative, and yet

publicly available simulator, of this kind is the Chaos Router Simulator [11], targeted at the chaos router and oblivious routing, with various traffic patterns for both mesh and torus network.

An event-driven simulator is very slow because of enormous software overhead, although it may be more accurate than cycle-driven. NETSIM (General Purpose Interconnection Network Simulator) [41], ICNS (Interconnection Network Simulator) [81], JANET (A Flit-Level Multiprocessor Network Simulator) [47] and PP-MESS-SIM (Flexible and Extensible Simulator for Evaluating Multicomputer Networks) [62] are all representative of work in this field. This kind of simulator is generally limited to a small design space because of the speed.

The analytical simulator is ideal, both fast and accurate, but it is very hard, if not impossible, to model a large complicated network with a large number of options. The use of this model is limited in the analysis of the queue effect or simplified case of specific design space [57].

Each of these simulators has its pros and cons, and all of them designed so far are limited to a small design space. For example, although cycle-driven simulators are more likely to be used for exploring large numbers of design tradeoffs, none described in the literature takes into account tradeoffs of router arbitration policies, VCT and WH routing, dynamic and static balance in the same platform.

2.3.3 The Cost-Speed Model

The objective of cost-speed evaluation is to get the implementation cost of a specific set of design parameters so that the performance evaluation results finished by flit-level simulator can then be corrected to best describe the real scenario.

A significant work in this field is the cost and speed model for k-ary n-cube wormhole routers [16]. Most of the necessary parts are modeled into a unified (canonical) router architecture that satisfies all common features of the domain. The relationship between the cost-speed and the component parameters was derived from the design of the PAR [4]. The arbitration policy used was Direct-First. The functions are cited and listed in Table 2.1 for mapping into a 0.8-micron gate array.

Component	Parameter	Gates	Time Delay
RG: Routes Generator (Address Decoder)		O(1)	2.70
PS: Path Selection (Arbitration)	F (freedom)	$O(F^2)$	0.60 + 0.6 * logF
HU: Header Update	F (freedom)	O(F)	1.40 + 0.6 * logF
CB: Crossbar Switching	P (ports)	$O(P^2)$	0.40 + 0.6 * logP
VC: Virtual Channel Controller	V (# of VCs)	O(V)	1.24 + 0.6 * logV
FC: Flow Control		O(1)	2.20

Table 2.1. The Cost-Speed Model mapped into a 0.8-micron gate array tech. Note: P is the number of inputs/outputs of crossbar (P x P), F is the number of output choices an input can have (typically same as P), and V is the number of VCs multiplexed into one physical channel.

The router contributes to the network performance issues of latency and throughput by determining routing setup delay and flow control delay. The routing setup requires

$$T_{rs} = T_{RG} + T_{PS} + T_{HU} + T_{CB} + T_{VC}$$
$$= 6.34 + 0.6 * (\log P + \log V + 2 * \log F) \text{ ns},$$

while the flow control requires

$$T_{fc} = T_{FC} + T_{CB} + T_{VC}$$
$$= 3.84 + 0.6 * (\log P + \log V) \text{ ns}.$$

The final router cycle time is decided by

$$T_r = \max (T_{rs}, T_{fc}).$$

The model was used to evaluate the cost-effectiveness of adaptive wormhole routing [30]. The setup time for a header was assumed to use two stages (cycles), finish the path selection, and then proceed through the switch. External delay that contains the virtual channel multiplexing time was also considered. The path setup time contains three parts,

$$T_p = T_{RG} + T_{PS} + T_{HU}$$
$$= 4.70 + 1.2 * \log F.$$

The time to cross the switch contains an extra 0.8 ns setup time for output channel latch (T_{SU}),

$$T_s = T_{FC} + T_{CB} + T_{SU}$$
$$= 3.40 + 0.6 * \log P.$$

An extra 4.9 ns is assumed for a flit across the physical of-chip wire and be latched onto downstream input channel (T_{PC}), besides the virtual channel control delay,

$$T_c = T_{VC} + T_{PC}$$
$$= 6.14 + 0.6 * \log V.$$

When pipelined, the above 3 operations must be performed in 1 clock cycle, so the clock period is

$$T_r = \max(T_p, T_s, T_c).$$

Although the cost-speed model is very useful to estimate the relative merits of design tradeoffs, it is not universally applicable. First of all, not only the absolute constant factors, but also the relative values among them may change. Due to the progress in VLSI technology, channel propagation delay, rather than control complexity becomes more critical, so that it favors adaptive schemes. Different partitions of a design may lead to quite different optimization, which will affect the derived constant [76]. Therefore, to evaluate a system by just synthesizing its components is in itself problematic. Second, the model itself is derived from a non-pipelined version, any further pipelining or parallel processing will not only add gates, but also change the timing. Finally, the model itself does not take into account all possible tradeoff parameters. For example, the size of the buffer that we are interested in for buffered wormhole routing is not modeled, and also the arbitration policy is not starvation-free.

3 DYNAMIC-BALANCED TORUS ROUTING

This chapter begins with an optimization scheme for virtual channel static balancing and then introduces a dynamically balanced scheme to enhance the performance and the flexibility of previous routing algorithms.

3.1 Static Virtual Channel Balance Optimization

Before presenting the dynamically balanced algorithm, we adapt the basic ideas of the T3D to our needs. First, we model both the static and the dynamic schemes in a unified architecture in which no inter-dimension channels are used. Second, because there are no publicly available tools for T3D optimizations, we build a general optimization package so that more results can be generated and used for further analysis.

3.1.1 Optimizing Mechanism

To model statically and dynamically balanced algorithms in the same canonical router architecture presented in the next chapter, several changes have been made to the original T3D scheme:

1. Inter-dimension channel selection is done dynamically and based on availability. Since the inter-dimensional channels do not form a ring, this does not alter the deadlock properties of the network but it does improve the throughput [36].
2. To balance the use of virtual channels, we try several random combinations of optional assignments to unconstrained routes until there is no significant improvement in the balance. The optimization is done globally, while it is done either globally or hierarchically in T3D.
3. Our canonical model works on an abstract level because more design tradeoffs than those in the T3D will be explored. The node in the model can be a single PE, a SMP, or even a low level sub-network, while in the T3D two PEs share the same router interface.

4. For the same reason, the separated request network is removed so that only two VCs are left for each PC link.

Actually only the first and the second change will differentiate the optimization here with the T3D.

The basic ideas in T3D, the concept of dateline shown in Figure 2.4; the imbalance cost [63] is adapted into our optimization package. A VC imbalance is defined by dividing the absolute difference in VC1 and VC0 traffic by the sum of the two, thereafter a perfectly balanced link has an imbalance of 0. To penalize the highly imbalance link while speeding up the optimization, the total imbalance cost, which is the sum of squares of the VC imbalances, is used in the package.

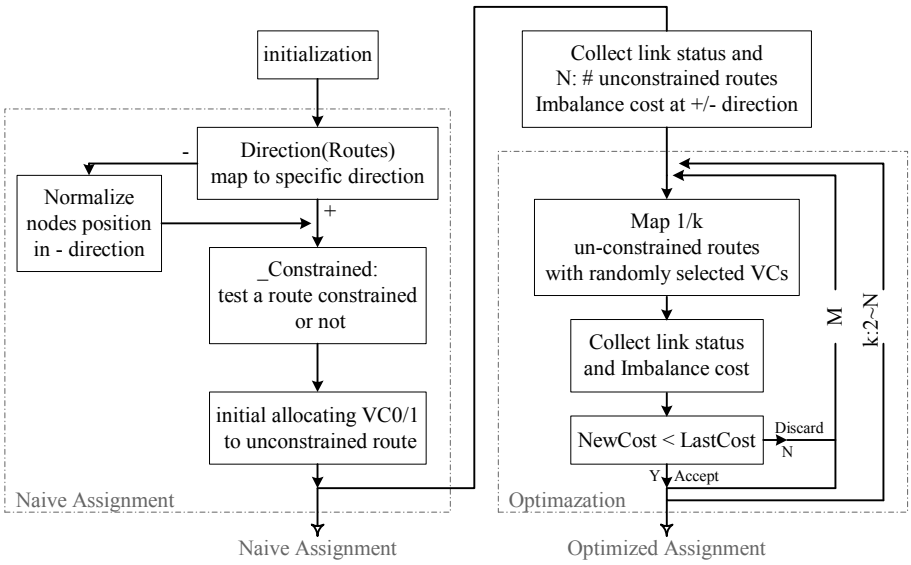


Figure 3.1. The block diagram of the static optimization of VC assignments.

The flow chart of the software is shown in Figure 3.1. The optimization is based on a naïve VC assignment, which is equivalent to what the TRC does. A route is mapped in a specific direction in a bi-direction ring according to the shorter path to its destination. If the route length equals to half of the length of the ring, it is mapped in such a way that all similar routes are mapped evenly in both directions. Whether a route is constrained or not depends on whether it will cross the dateline. As the dateline is set on different nodes for each direction, the node position is normalized so that the common functions can be used for each direction without noting the difference. To speed the optimization process, more unconstrained routes are picked out and assigned randomly at the beginning than in the following tries.

To visualize the balanced assignment of VCs, the global optimization results for rings with 8 and 16 nodes are illustrated in Figure 3.2.

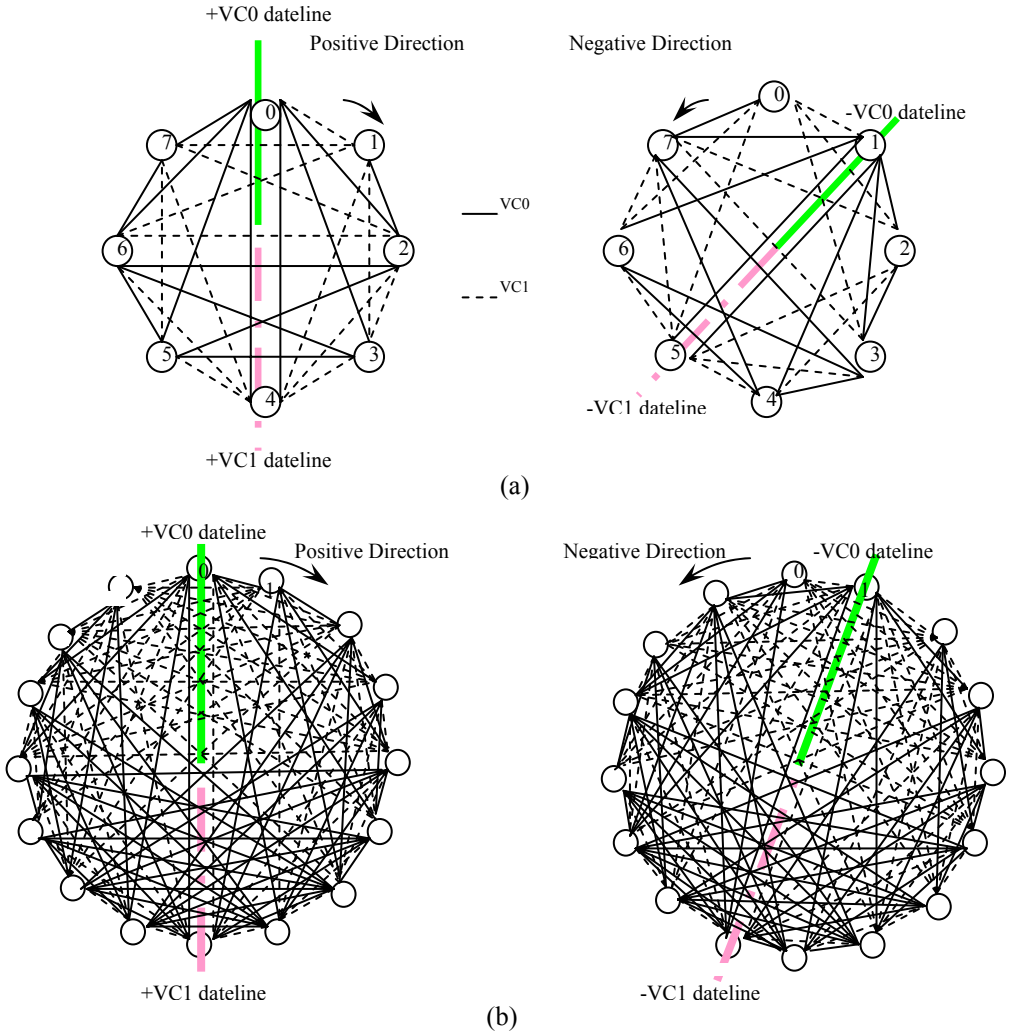


Figure 3.2. The Static VC Balance Optimization. (a). The optimal VC assignment for width 8 ring. (b). The optimal VC assignment for width 16 ring.

3.1.2 Static Balanced Routing

The VC balance optimization works on a bi-directional single ring. To route on a multi-dimensional torus, dimension order routing (DOR) is used to guarantee that it will be deadlock-free and a built-in lookup table is used to balance VC usage for routing inside the rings of each dimension.

Table 3.1. The lookup table for statically balanced routing on an L-node long bi-directional ring.

VC cur \ dest	0	1	…	**d**	…	L-2	L-1
0	0	C0P		C1M		C0M	C0M
1	C0M	0		C0M		C0M	C0M
…							
c	C1M	C0M	…	**C0P**		C0P	C1M
…							
L-2	C0P	C1P		C1M		0	C0P
L-1	C0P	C1P		C1M		C1M	0

The lookup table for a ring with length L is illustrated in Table 3.1. CxP (or CxN, x = 0,1) represents the VC selected for a route that cannot cross the VCx dateline in the P (positive, or N: negative) direction. For example, the route from node c to node d should use the CoP virtual channel. The algorithm for the multidimensional torus, ***T3D-like*** (or ***StaBal)***, can be implemented as:

- If dest_node_addr == cur_node_addr, then the packet reaches its destination and should be drawn off the network through the ejector, so dest_channel = ejector (0);
- However, if dest_node_coordinates[cur_dimesion] != cur_node_coordinates[cur_dimension], then the packet should proceed along its current direction in current dimension using the current virtual channel, so dest_channel = cur_channel;
- Otherwise, the packet has finished routing in this dimension and should change its path to the next higher dimension using an updated channel, so dest_channel = 4 * next_dimension + LOOKUP_TABLE[cur_node_coordinates[next_dimension][dest_node_coordinates[next_dimension]].

Another implementation of the static-balanced routing embeds the VC assignments in the packet header when it is injected from local node to the router interface. This will shift the cost of the lookup table from the router to its local node, where it can also be implemented in software. This will make the router

faster. The disadvantage of this strategy is that it not only adds overhead to the packet, but also, more importantly, expands the phits.

3.1.3 The Problems of Static Balance

The drawbacks of static balance, as discussed in Section 2.2.3, still exist after the adaptation of the original T3D scheme:

- The optimization is based on a uniform workload, meaning that it may perform poorly under non-uniform traffic, which is possible to our system to do FFT or matrix transform.
- Only one specific optimized assignment can be used, although multiple choices exist. For example, if optimized hierarchically, the assignments may lead to degraded performance if globally uniform traffic is applied.
- When the network needs to be reconfigured for a larger system, the assignments must be re-optimized and the lookup table must be refreshed.
- The VC assignment will be very hard to balance when the network width gets bigger, as listed in Table 3.2.

Table 3.2. The relation between the imbalance of static assignment optimization and the ring length.

Ring Length	4	8	16	32
Positive Link Average Imbalance	0	.0625	.2070	.2715
Negative Link Average Imbalance	0	.0625	.2109	.2700

3.2 Dynamic Balanced Routing

The ideas of static virtual channel balance and the cycle-tolerant deadlock prevention theory can be adapted into a more powerful routing algorithm. To facilitate the analysis, related definitions and assumptions are presented first, followed by proofs.

3.2.1 Definitions and Assumptions

The analysis is based on two definitions. The basic dateline concept is re-defined to remove the bottleneck formed in the original partial adaptive routing algorithm, Duato's PA or ParAdapt (see section 2.2.4). An escape channel distinguishes itself from a cyclic channel, uses the buffer space more efficiently, and still avoids the deadlock situation.

Definition 1: A **hardwired dateline** is a dateline, which cuts a ring so that it can only be crossed by traffic in the appropriate VC. It is known by each node on the ring. The position of the dateline is recorded as the node address where the dateline crosses the switch. It can be implemented by broadcast during network initialization, or by a buffer that takes effect after power-up. Comparing with Figure 2.4 and 2.5a, we can see that the hardwired dateline for a unidirectional ring in Figure 3.3 crosses the link where the bottleneck is formed in partial adaptive routing.

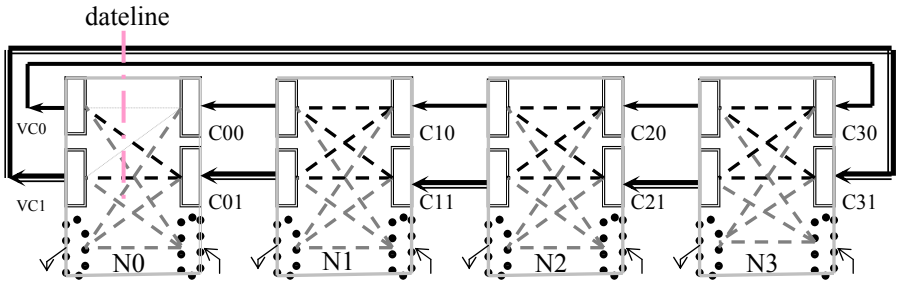


Figure 3.3. The hardwired dateline crossing a unidirectional ring.

Why should the dateline be hardwired? The reason is that we want to do the VC balancing dynamically and in a distributed manner, rather than by using static off-line optimization. The dateline can be viewed as the watershed for the two virtual channels, between the input and output buffer of VC0 and VC1, not for the injector and ejector channels because they are not involved in any cycles. There will be no real connection between the input buffer of C00 and the output buffers C31 or C30. That is, a packet that arrives in input buffer C01 can use either C30 or C31 to reach its destination so that the bottleneck is removed.

Definition 2: An **escape channel** is an open-loop channel, which consists of a set of VC buffers without a cyclic direct-dependency among them. The escape channel can be identified from Figure 3.3 or from its corresponding channel-dependency graph shown in Figure 3.4.

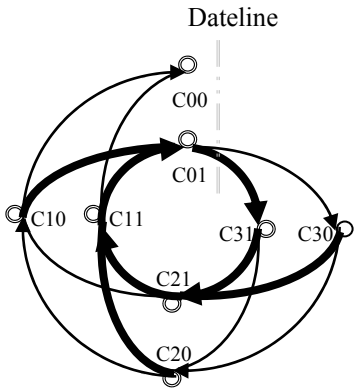


Figure 3.4. The channel-dependency graph.

The virtual channel VC0 consisting of the set of CX0 is open-looped so that it is an escape channel, while the virtual channel VC1 is a cyclic channel. The critical difference between the two kinds of channels is that routing on the former will never cross the dateline, while that on the latter may cross if a packet is heading to the right of its current position.

The following assumptions are used for further discussion; most are the same as those assumed by the cycle-tolerance theory:

1. A node can generate packets of arbitrary length and destined for any other node at any rate subject to the constraints of available source queue space.
2. A packet arriving at its destination node is eventually consumed.
3. Once a queue accepts the first flit of a packet, it must accept the remainder of the packet before accepting any flits from other packets.
4. The routing control unit of a receiving queue may arbitrate between packets that request it in a manner with no starvation but may not choose among waiting packets.
5. Packets follow minimal path and the routing function offers at least one minimal path.
6. A route taken by a packet depends on its destination and the status of output channel; routing therefore has some degree of adaptivity.
7. A hardwired dateline exists for each unidirectional ring of the network.
8. The queue in an escape channel can hold flits belonging to different packets, while the queue in a cyclic channel cannot.

The assumptions 1 ~ 5 are valid for TRC and Duato's PA. Item 6 is valid for the latter but invalid for TRC where the route is decided only by packet's destination. Item 8 is different from what is made in Duato's PA where all queues cannot hold flits of different packets [26].

3.2.2 The Algorithm

Let us first present the algorithm, ***DynBal***, for routing in a single unidirectional ring and then adapt it to a bi-directional ring and finally to a multiple dimensional torus. Routing in a single ring, as depicted in Figure 3.3 with VC1 (CX1) as cyclic channel and VC0 (CX0) as escape channel, can be stated as follows:

- *If a route crosses the hardwired dateline, before crossing it, select VC1, the cyclic channel (CX1) only;*
- *Else, the route does not cross the dateline or has already crossed the hardwired dateline, so select VC0, the escape channel (CX0) with high priority, or VC1 with low priority.*

The requests to the cyclic channel can be granted if the output queue is empty and those to the escape channel can be granted if the output queue space is available, so as to satisfy assumption 8. By doing this, only packets not crossing the hardwired dateline can hold and co-exist in a channel buffer CX0 while all packets heading to the right of their current position can only take a buffer CX1 without any co-existence with others.

If the routing algorithm is deadlock-free for a unidirectional ring, which will be proved true in a following section, it is easy to adapt to a bi-directional ring, as shown in Figure 3.5. To break the tie between the rings in opposite directions, while abiding by the same convention used by the previous routing algorithms, a packet is routed to a specific directional ring at the source node, and keeps proceeding in the same ring until it reaches its destination, where it is drained-off. There are no dependencies between the rings in opposite directions, so that a packet in one ring never blocks a packet in the other ring. The selection of rings in the source node is based on the shortest path, and there are two different datelines for each ring.

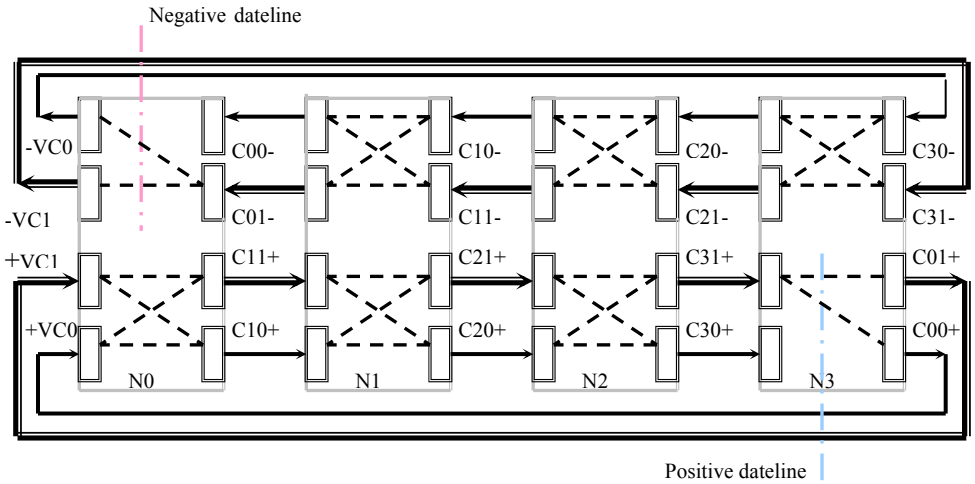


Figure 3.5. The dynamic balanced routing in a bi-directional ring.

Just as shown in the partial adaptive algorithm, crossing dimensions on a multi-dimensional torus should use the dimension order routing (DOR), and in each dimension routing the packet along a particular ring should follow the rules of the above algorithms. So, the virtual channel balance is only within each ring rather than globally for the whole network.

3.2.3 Freedom from Deadlock

The proof of deadlock-freedom of the dynamic balanced routing algorithm begins by decomposing the torus network into its component unidirectional rings. The rules of crossing hardwired dateline make the extended channel dependency graph acyclic, and the rules of packet-co-residence in acyclic escape channel rather than cyclic channel do not lead to deadlock or starvation. If minimal source routing is used to split the tie of two unidirectional rings in a bi-directional ring, and DOR is used to cross multiple dimensions, then the algorithm is still deadlock-free for the bi-directional ring and the multi-dimensional torus.

Theorem: The routing algorithm, DynBal, is deadlock-free.

We begin with the analysis of the hardwired dateline we added to the original *Duato's PA* algorithm, without considering the packet-co-residence in escape channels. That is, we use the exact same assumptions adopted by *Duato's PA*.

Lemma 1 The *DynBal* routing algorithm on a single uni-directional ring embedded with hardwired dateline, while not allowing packet-co-residence for all buffers, is deadlock-free.

Proof sketch. As this lemma is based on the exact same assumptions of the *Cycle-Tolerated Theory* cited in the section 2.2.4, what we need to do is to find a routing sub-function R1 for DynBal, which should be connected and have a non-cyclic extended channel dependency graph.

Lemma 1.1 TRC is a routing sub-function, R1, of DynBal, and is connected.

Proof of lemma 1.1. Comparing Figure 2.3b with Figure 3.3, R1 equals to R except that C21 and C30 cannot be used and C01 can only be used to forward a packet to its destination higher than the current node. R1 is connected because a packet at node Ni destined for a lower node will be forwarded through Ci0 and a packet destined for a higher node will be sent across Ci1. □

Lemma 1.2 The extended channel dependency graph of R1 is acyclic.

Proof of lemma 1.2. Although the channel dependency graph of DynBal, shown in Figure 3.4, contains several cycles, the extended channel dependency graph for routing sub-function, R1 of DynBal, depicted by the dark lines in Figure 3.6 has no cycles. Compared with Figure 2.5c, it just introduces more indirect dependencies than *Duato's PA*. □

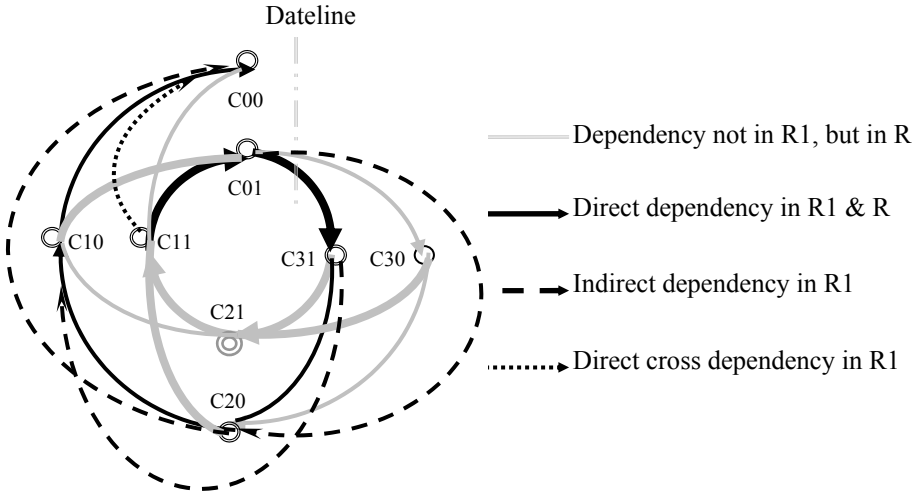


Figure 3.6. The extended channel dependency graph of routing sub-function R1 of DynBal.

Proof of lemma 1. It follows immediately by lemma 1.1 and 1.2. □

Several observations of lemma 1 are listed bellow:

- It is the introduction of the hardwired dateline that leads to more dependencies and thereafter more flexibility; the bottleneck in *Duato's PA* is removed. The dateline here is no longer an imaginary line, it must be implemented in hardware as a reference to see whether a packet crosses it or not. The dateline can be arranged to cross any nodes in the ring, but each node knows where the dateline is. For a packet entering into the input buffer, the router must decide the path to reach its destination, and must check whether the path will cross the dateline node or not before moving it to a proper output buffer.
- There are four different ways the virtual channels are involved in routing. The first, {Cx1 -> Cx1}, is the set of buffers along VC1 that is used by the routes crossing the dateline, and forms a channel dependency cycle. The second, {Cx1 -> Cx0}, is the set of escape channels from {Cx1} to {Cx0} that is used when there is any temporary blocking in the cycle. The third, {Cx0 -> Cx0}, is the set of buffers along the escape channel that is used by the route not crossing any dateline and forms no cyclic dependency. The last, {Cx0 -> Cx1}, is the set of routes that can be used to switch a packet already in {Cx0} to {Cx1} if the former is more congested and the latter is free at a specific time.
- The packet in the cyclic dependent channel can be finally released, because:

  a). For the {Cx1 -> Cx1} cyclic dependent channels, there exist escape channels {Cx0} to release a packet to acyclic dependent channels {Cx0 -> Cx0} after the packet has crossed the dateline, while the packet in the acyclic dependent channels {Cx0 -> Cx0} will be finally released after a limited time.

  b). Although there is another kind of cyclic channel dependency, {Cx0, Cx1}, the priority scheme guarantees that the oscillation will be released in a finite time for two reasons. First, there is at least one escape channel from each of the dependent cycles to the acyclic dependent route, so it is impossible that the packet will be stuck in one of the cycles for an un-limited time (deadlock.). Second, the higher priority is always given to the escape channel -- this not only makes the packet in the cyclic route proceed as soon as possible, but also eliminates unnecessary switching between cycles. If the packet escaping from a cycle can proceed in a cyclic route, it will do so and release the dependent cycle, even though it may return to the cycle (form the dependency

between cycles) again because of a channel's availability. If both the cyclic and the acyclic routes are unavailable, then the packet will wait until its precedence has proceeded. It is guaranteed that the packet in the cyclic channel will be finally drained out in a limited number of cycles because of both the existence of escape channels and the priority adopted.

Further, from lemma 1, it can be shown that before accepting the header of another packet, it is not necessary to empty the queue along an escape channel, but it is mandated to drain out the queue along a cyclic channel.

Lemma 2 Without causing deadlock in DynBal, multiple packets can co-reside in a queue of the escape channel, but cannot co-reside in a queue of the cyclic channel.

Proof sketch. We first show the logical necessity of packet non-co-residence for cyclic channels and then show that the co-residence is not a problem for escape channels.

Lemma 2.1 To be deadlock-free, multiple flits belonging to different packets cannot co-reside in the same buffer along a cyclic channel.

Proof of lemma 2.1. Figure 3.7 shows a configuration where multiple packets hold the same cyclic channel buffer and all packets head to a node two hops away.

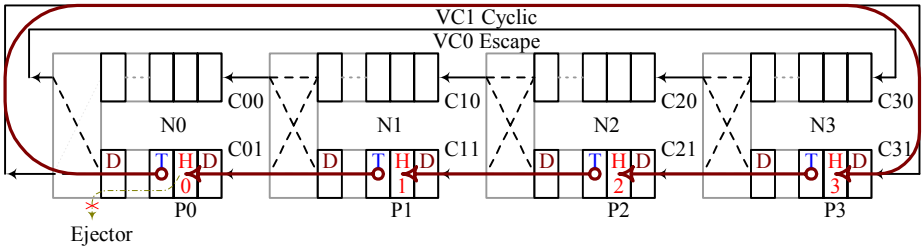


Figure 3.7. A deadlock configuration in a cyclic channel.

As none of the headers appears at the head of the buffers, no packets can be routed to an optional path, even though it may exist. For example, packet P2 in channel C21 heading to the node N2 (header H2 represents that the destination address is node 2) is permanently blocked by another packet, P1. The optional buffer C20 would have been selected if C21 were checked first when routing in C31.

Lemma 2.2 Any packet on hold in the escape channel must not cross the dateline.

Proof of lemma 2.2. This is follows from the rules of DynBal and it is the basis for further analysis. The configuration shown in Figure 3.8 is illegal because the packet in virtual channel C01, P0, heading to

node N3, must cross the dateline. When transferred from N2 to N1, it should not take the escape channel C10. Packet P1 can be switched to a cyclic channel C11 from C20, which is another optional channel. On the other hand, packets P3 and P4 are both legally routed to the escape channels, C20 and C30, from the cyclic channel C31 and C01, respectively.

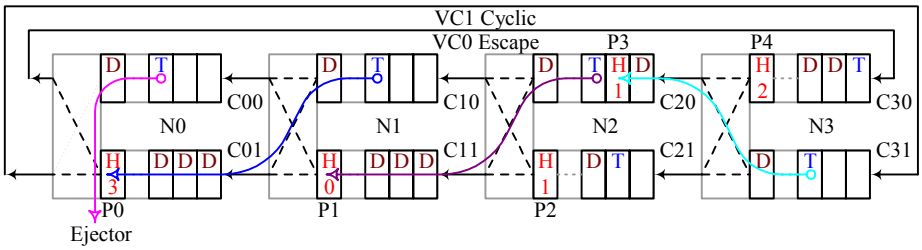


Figure 3.8. An example of illegal configuration of channels.

Lemma 2.3 Packets in an escape channel will never be blocked indefinitely.

Proof of lemma 2.3. There are two cases where the escape channel is involved in the routing process: 1) All packets in the escape channel head straight to their destinations via the escape channel itself, and 2) some packets use both the escape channel and the cyclic channel to proceed.

The first case does not form deadlock because the escape channel is open-loop by definition, and any packet that does not cross the dateline by lemma 2.2, will eventually get to its destination.

The second case is more complicated because a cycle is formed. If a packet is switched to a cyclic channel where no other packet resides, it can be switched back to the escape channel, as long as it has not reached its destination. This is again true because of lemma 2.2: the packet must be heading somewhere without crossing the dateline. If the packet switched into the cyclic channel has arrived at its destination, it will be drained out into the ejector and the following packet will proceed further. Otherwise, as its header always takes the first place of the buffer, it can request the escape channel again. If all the packets, in and before the requested buffer along the escape channel, head straightly as in case 1, they will proceed and the current request is granted. If any of them has requested or been switched into the cyclic channel and been blocked, possibly because a cyclic dependency has formed along it, then the packet nearest to the dateline will proceed first. This is because the escape channel is always open-ended and the upstream end can only hold the packet headed to its local ejector. The packet P1 in Figure 3.9 headed to N0 can get its turn to use

C00 after packet P3 releases it. If there is another following packet temporally blocked by P1, it can proceed thereafter.

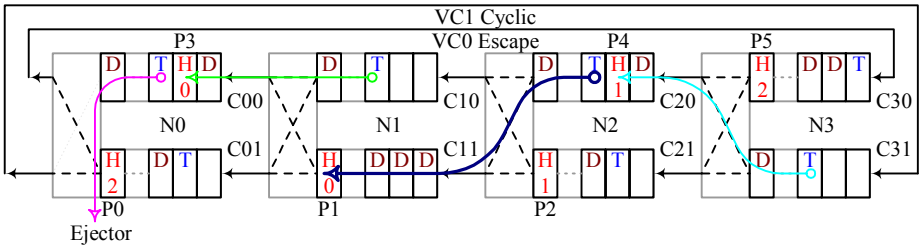


Figure 3.9. The dependency between the cyclic and the escape channel.

Lemma 2.4 Packets in a cyclic channel will not be blocked indefinitely.

Proof of lemma 2.4. Because the packet header always takes the head of the buffer, it can have three options to proceed:

1) Be drained out locally if the local node is its destination.

2) Be switched into an escape channel if the path to its destination does not cross the dateline.

3) Be passed straight into the cyclic channel, which is necessary if it does cross the dateline.

Options 1 and 2 never form deadlock, which is guaranteed by lemma 2.3. The worst scenario of the last option is illustrated in Figure 3.10, where each node sends a packet to a node two hops away.

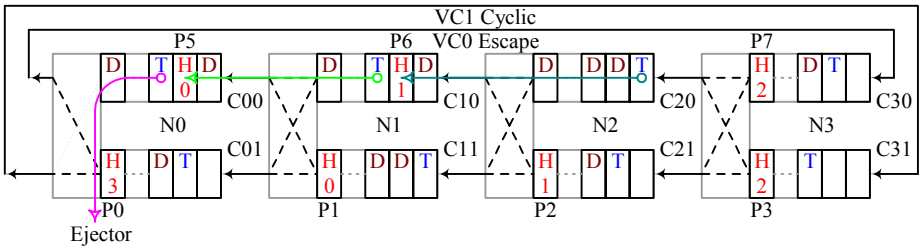


Figure 3.10. The cyclic dependency formed in the cyclic channel and its tearing up.

If only the escape channel is used, the scenario would be the same as that is shown in Figure 2.3a – a deadlock. According to lemma 2.3, the escape channel will be released in a limited time and the cycles formed in the cyclic channel will be torn up afterwards by switching a packet into an escape channel. For example, packet P5 will be drained out off C10 after its precedent being ejected locally, and then C00 is

released. Then, either packet P1 in cyclic channel C11 or packet P6 in escape channel C10 can proceed via C00. If P1 gets its turn first, the cycle is broken in C11 so that packets P2, P3, and P0 can proceed after all flits preceding them are out of C11, C21, and C31 respectively. If P6 gets the turn first, the cycle is broken at C31 and packet P3 can be switched into C20 without waiting for the empty of C20, and then P0, P1, P2 can be forwarded along the cyclic channel in order. □

Proof of lemma 2. It has been proved by lemma 2.1 that non-packet-co-residence in the cyclic channel is necessary for deadlock-freedom. Following the rules of DynBal, the packet-co-residence in an escape channel will result in deadlock by lemma 2.3 and lemma 2.4. □

Proof of the theorem. By lemma 1, the dateline embedded in the system makes the routing deadlock-free if the rule of non-packet-co-residence is applied to all kinds of buffers. By lemma 2, the non-co-residence is only necessary for the cyclic channel. Consequently, the DynBal algorithm is deadlock-free following the assumptions made in Section 3.2.1. □

3.3 Fully Adaptive Routing

The DynBal algorithm adopts adaptivity to perform dynamic balancing within a ring. Fully adaptive routing on multi-dimensional torus is achievable by introducing another virtual channel, the ***fully adaptive channel***, similar to Duato's FA or T3E depending on how the extra VC is used.

3.3.1 Dynamic Balanced Fully Adaptive Wormhole Routing

For convenience of analysis, we begin with several definitions.

Definition 3: A **fully adaptive channel** is a virtual channel in which a packet can change its dimension at any time.

Definition 4: A **non-fully adaptive channel** is a virtual channel in which a packet can change its dimension only after finishing the routing in its current (lower) dimension.

Definition 5: A **virtual network (VN)** is a sub-network constructed by some of the virtual channels in the original physical network.

Definition 6: A **fully adaptive virtual network** is a virtual network constructed exclusively by fully adaptive channels.

Definition 7: A **non-fully adaptive virtual network** is a virtual network constructed exclusively by non-fully adaptive channels.

If a third virtual channel is introduced to DynBal as a fully adaptive channel, by the similar method as used in Duato's FA, a fully adaptive wormhole routing algorithm, ***F_DynBal***, can be defined:

- *Routing in the lowest of the dimensions not yet finished can use either the non-fully adaptive VN or the fully adaptive VN. The fully adaptive channel can be used the same way as a cyclic channel in each of the rings of this dimension where the DynBal is applied;*
- *Routing to a higher than the lowest dimension of the dimensions not yet finished can use only the fully adaptive VN.*

It should be noted that the fully adaptive channel would form a cycle by crossing dimensions (the four-corners), but it is different with the cycle formed in a single ring (the cyclic channel).

The process of proving that the F_DynBal is deadlock-free is quite similar to what has been done for the DynBal, but with a much more complicated extended channel dependency graph. This can be simplified, however with the two scenarios it is built on. These are now described.

We begin with a modification of the DynBal algorithm by introducing another cyclic channel into the single ring, as shown in Figure 3.11. An intermediate routing algorithm is defined by treating the VC2 exactly the same as the VC1. Any packet crossing the hardwired dateline should use VC1 or VC2, while the packet not crossing the dateline can use any one of the three virtual channels. The extra channel added does not change the fact that the escape channel is open-looped. The cycle formed by VC1, VC2, or any two of the three, can be broken up by switching the packet to the ejector (if done) or to an escape channel (if continuing on).

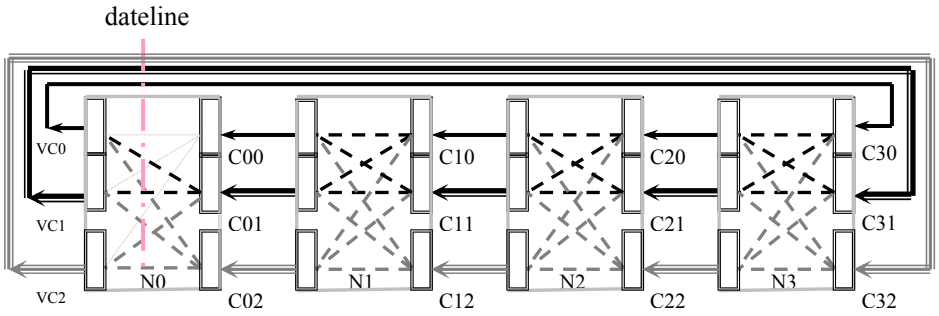


Figure 3.11. The third virtual channel added to a ring for DynBal routing.

Because arbitrarily crossing dimensions can only be achieved by routing in a fully adaptive VN, only fully adaptive channels can construct the cycle formed by the four turns, or corners. This scenario can be transformed to a similar picture as shown in Figure 3.11. The four-turn cycle is escorted with couples of the escape channel and the cyclic channel in each dimension, which are always the exhaust if cyclic waiting happened in the fully adaptive channel. ??

Since either the cycle added to the ring or the cycle formed by four-turns does not lead to deadlock, the algorithm F_DynBal is deadlock-free. Figure 3.12 shows an example of routing on a bi-directional 2D torus.

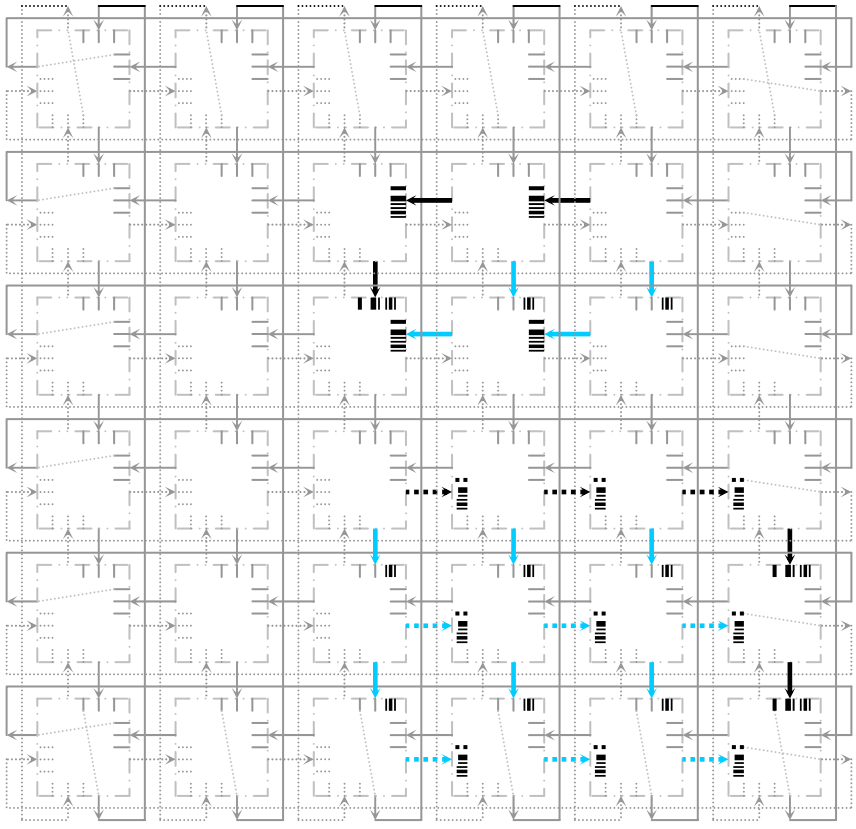


Figure 3.12. An example of F_DynBal routing on a bi-directional 2D torus.

The algorithm is expected to provide a better performance than DynBal especially for a non-uniform traffic, and is also an improvement over Duato's FA as the bottleneck is removed and buffer space is used more efficiently.

3.3.2 Hybrid Dynamic Balanced Fully Adaptive Routing with Drain-Off Buffers

The fully adaptive channel in F_DynBal or Duato's FA is restricted to satisfy the assumption that it cannot hold multiple flits of different packets. It introduces bubbles to the link, especially as the ratio of buffer size to packet size increases.

If the maximum packet size is limited, and the fully adaptive channel (buffer) can guarantee enough space to drain a whole packet out of another channel, the buffer is referred to as a ***Drain-off buffer***, and the fully adaptive VN is also referred to as a ***Drain-off VN***. A new algorithm, ***FD_DynBal***, can be defined as:

- *Routing in the lowest dimension of the dimensions not yet finished can use either non-fully adaptive VN or fully adaptive VN (the drain-off VN). The drain-off buffer can be used in the same way as a cyclic channel in each of the rings of this dimension where the DynBal is applied;*
- *Routing to a higher than the lowest dimension of the dimensions not yet finished can use only the drain-off buffer;*
- *The drain-off buffer must guarantee space for all flits of a packet following the accepted header.*

The algorithm is actually a hybrid scheme of VCT and WH, similar to what is used in T3E. It is deadlock-free based on two facts. First, no cyclic dependencies are formed by switching packets from any channels in a non-fully adaptive VN to a drain-off VN simply because the nature of the buffer space is guaranteed. Second, the cyclic dependency among drain-off buffers is possible, but will always lead to a simple case that each buffer involved will have a packet header at the head of the buffer (although there are other packets following and co-residing in it). In this case, packets can be routed to the non-fully adaptive VN and be guaranteed delivery, in the same way as in the DynBal routing algorithm.

By using the statically balanced scheme and simplifying the original T3E network by removing the request-response network and using the dimension order routing instead of a direction-order, another algorithm, ***T3E-like***, can be derived. It is also deadlock-free. The only difference between the FD_DynBal and the T3E-like is the routing inside the non-fully adaptive VN: the former uses DynBal while the latter uses T3D-like.

The differences between drain-off buffer based routing and the fully adaptive wormhole routing are the buffer space requirement of the fully adaptive channel and the way it is handled. The fully adaptive channel in wormhole routing does not guarantee space for a whole packet, while this is required in the drain-off buffer-based hybrid method. A fully adaptive channel is not allowed to hold flits belonging to different packets, while this is allowed in the hybrid scheme. These two kinds of algorithms are also different in the system that they are applied to. A general fully adaptive (WH) routing algorithm is suitable for a variable-packet-sized network without limit to the maximum packet size. The hybrid scheme can be used either in a fixed packet size network, or a variable packet size network with known maximum packet size. Also, the drain-off buffer should be big enough to hold at least one packet. Even under the common domain, it is critical to evaluate which scheme is better.

4 PERFORMANCE EVALUATION

To answer the questions presented in sections 1.2.1 to 1.2.3, the router and the network must at first be modeled with reasonable assumptions. Following a description of the evaluation tool, a flit-level cycle-driven simulator, some key issues about routing algorithms under various network design tradeoffs and workloads are discussed.

4.1 Simulator Modeling

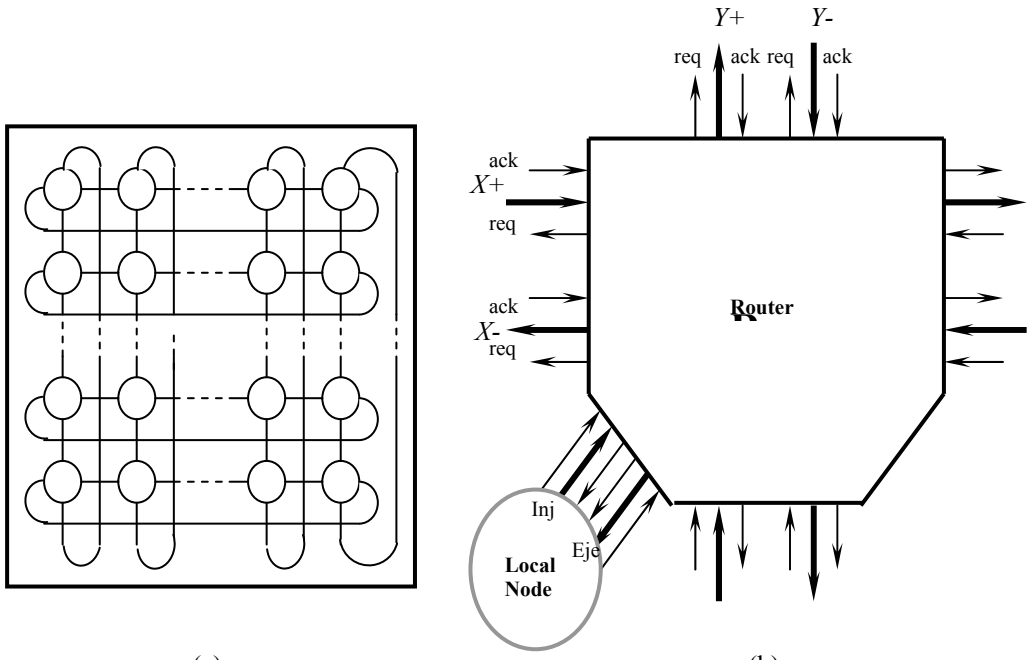
The simulator is modeled according to common abstract network characteristics rather than specific implementation details. The simulator is maintainable and flexible for further extension.

4.1.1 Canonical Router Network Model

The following discussion is based on the architecture shown in Figure 4.1. Several basic features of the model are listed below:

- The network is either a unidirectional or bi-directional torus where each node can communicate with each other node by an attached router.
- The node is not necessarily a single PE; it can be a local network in any topology, for example, a cluster of workstations [6][13][35][38][39], or a number of PEs connected by a bus. In any case, there must be an input (injector) and output (ejector) port to communicate with the router.
- Buffering at both input and output is modeled inside the router [26][45][78].
- Each virtual channel can be constructed with several lanes for performance reasons rather than for deadlock prevention [36].

- All virtual channels (and the lanes constructed within them) in the same direction share the same physical channel, unless specified otherwise.
- The width of the physical channel (phits) is assumed to be equal to that of flits.



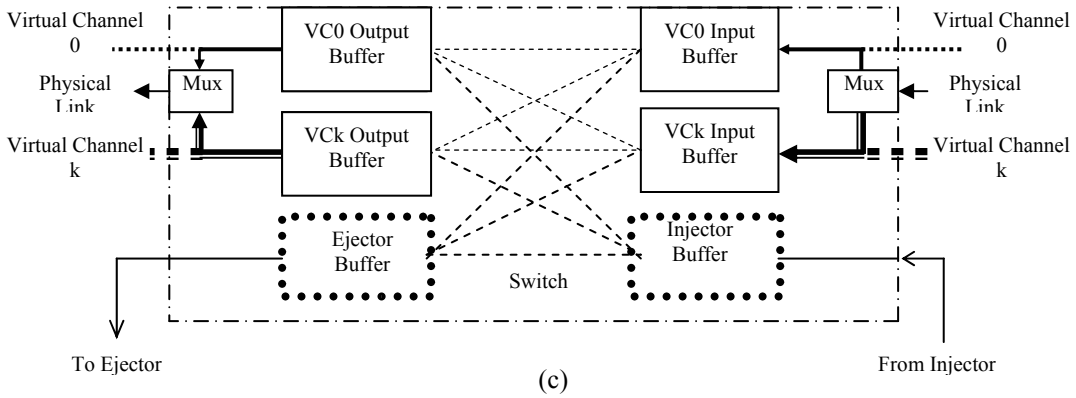


Figure 4.1. The canonical model of router networks. (a). A torus network. (b) A node with a router attached. (c). The details of buffering and switching for k virtual channels and injector/ejector inside the router.

The model is abstract [70]. For example, no specific number of virtual channels is specified because it is algorithm dependent and many algorithms will be evaluated using the same platform. It is also easily extendable. For example, the canonical model is generally based on the assumptions discussed below.

Canonical model assumption 1*: The network is tightly coupled in such a way that the physical link wire delay will not become critical; therefore, phits transferring from an output channel to a downstream input channel will take only one cycle.*

If the network is not tightly coupled, the wire delay (x cycles for example) on a pipelined channel can be easily built into the canonical model by enlarging the input buffer, which must be larger than 2x, and tailing the output buffer with a x-unit shift register. The request to the output channel and the transmission from input to output buffer is dealt in a regular way -- flits are moved into the output buffer directly, and shifted one by one into tailing buffers to emulate the exact delay cycles. If the space left in the downstream input buffer equals to 2x, the transferring from the output buffer to the shift register should be stalled, while the flits already in the register can be accepted into the input buffer.

The network is characterized by several parameters:

- Topology. Torus is assumed in this dissertation, although other topologies, such as mesh, are also modeled in the simulator.
- Wiring. The torus is either uni-directional or bi-directional. Bi-directional torus is presumed in the following discussion unless explicitly specified.
- Number of Virtual Channels.
- Number of Lanes. A virtual channel may have more than one lane expected to enhance performance.
- Physical Channels (PC). All virtual channels and their corresponding lanes in one direction share the same physical link.
- Total Node Buffer Size. To fairly compare different schemes, total buffer space per node is used, rather than using space per lane, per VC or per PC.
- Ratio of I/O Buffer size. It is used to explore the effect of different space allocations to the input and the output buffer.

- Ratio of Drain-off Buffer Size over Packet Size. It is used exclusively for drain-off buffer based algorithms.
- Arbitration Policy. It is the policy to resolve conflicted requests for an output buffer or a physical link, such as Round-Robin, Random, and Highest-First.

4.1.2 Workload Generation and Consumption

The message unit of workload is a packet that consists of a header flit containing the destination node address as well as other control bits, data flits, and a tail flit. Various kinds of workload are applied into the network to monitor its behavior based on the following assumption.

Canonical model assumption 2: Packets are mapped to each node based on a global communication pattern, but packet flits are injected to an input buffer iff buffer space is available. A packet is consumed as soon as it reaches its destination because of an un-limited ejection buffer.

The reason to specify a global pattern is to describe the initial traffic distribution, and to make the evaluation process manageable and reasonable. The injection rate is automatically adjusted during the simulation to reflect the network status and load distribution. The un-limited ejection buffer is a reasonable assumption to evaluate the maximal throughput of the network because, otherwise, it may be throttled by the ejection rate.

From the standpoint of mapping packets to each injector, the workload can be mapped either dynamically or statically. A ***dynamic workload***, in which packets are generated at every cycle of the simulation, can be used to describe the performance of other design tradeoffs by a BNF. In a ***static workload***, on the other hand, the generation is done just once and is used to measure how many cycles are needed to deliver all the packets mapped. For example, a data matrix (e.g. a picture) can be mapped to the physical torus network where each node is mapped on a certain fraction of the matrix.

There are three categories of communication patterns used to map the workload:

- **Uniform Traffic**. Each node sends packets to any nodes with the same probability, or, each node has the same probability to communicate with a randomly selected node.

- **Non-uniform Traffic**. Some nodes have more opportunities to communicate with other nodes than the average. Typical patterns of it are defined in the simulator, such as Hot-Spots, FFT or Bit Reversal, Dimension Reversal or Image Transpose, Bit Vector, Shuffling, and Shifting.
- **Local Traffic**. All or most of the communications is restricted within some regions of the network, rather than globally distributed. In each region, the traffic may be distributed either uniformly or non-uniformly. The Random Near and Shifting are typical examples of local traffic.

Some of the communication patterns may need more specific parameters to describe. For example, hot-spots are defined by a pair (x, y) where x is the number of nodes that are more likely (than average) to act as packets' destinations, while y is the ratio of the bandwidth needs of hot-spots over that of average nodes.

4.1.3 Routing and Transferring

A flit takes one cycle to transfer from an input buffer to an output buffer or from an output buffer to a downstream input buffer if there is no congestion. A header flit entering into an empty input buffer will have to take one extra cycle to get the routing done. The routing for the header flit following the tail of another packet (note only if the algorithm for this VC permits packet co-residence) will be done parallel with the transmission of the flit ahead of it. If wire delay is considered, the transfer from an output buffer to an input buffer takes a number of cycles that is equal to the delay.

The routing algorithms evaluated are those discussed in Chapter 3, which are TRC, T3D-like, DynBal, Duato's PA, F_DynBal, Duato's FA, T3E-like, and FD_DynBal.

4.1.4 Flit-Level Cycle-Driven Simulator

The transmission unit in the simulator is simplified with flits rather than physical bits (phits) begin assumed because we model the router and the network in an abstract logic concept, rather than in hardware implementation details. The details, such as the physical link bandwidth (phits), and how many phits are contained in the flit, are concerned with the layout of the network and constrained by the technology used. The simulation is done in the cycle level rather than in the real time (ms, ns etc.) for the same reason.

The structure of the simulator, NetSim, is described in Figure 4.2. The core of program is written in C++ [70], while the user interface (parameter set up and results viewing window) is coded in Tcl/Tk [33]. A database is used to save user settings and evaluation results.

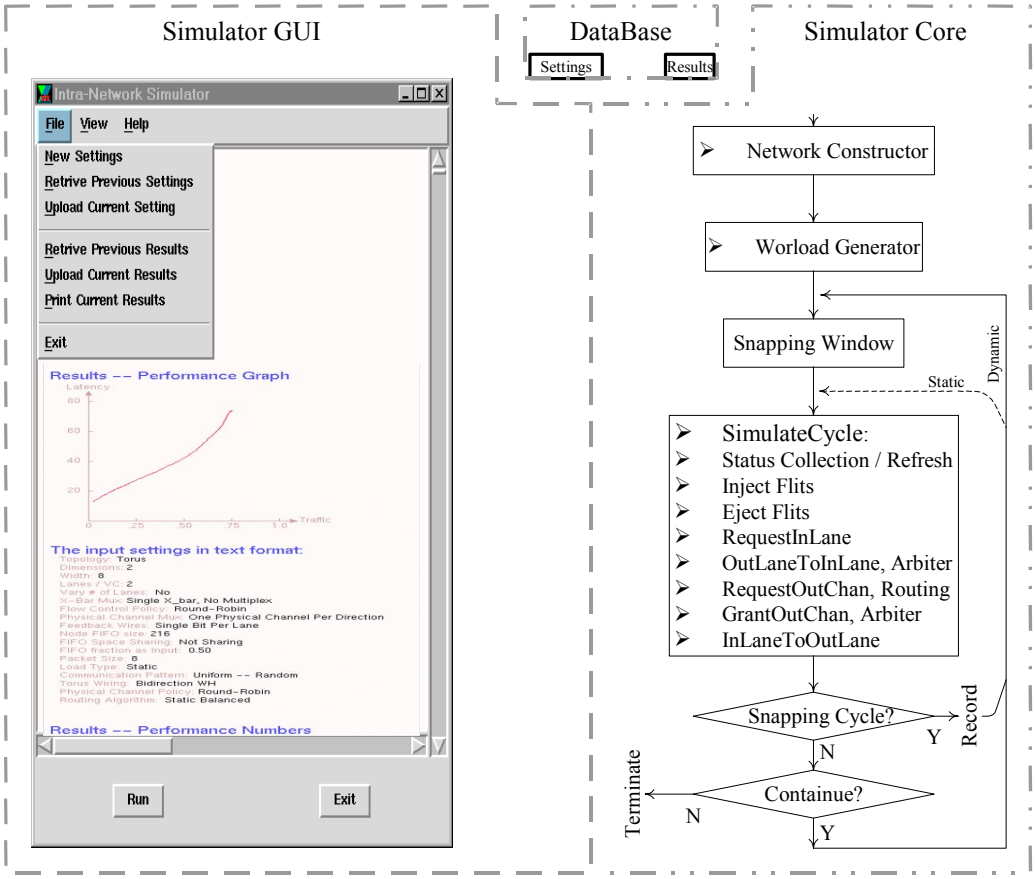


Figure 4.2. The NetSim, a flit-level cycle-driven simulator constructed by a GUI, a Core and a Database.

An experienced user can run the simulator either by command line input, or by retrieving the setup previously stored in the database. Users can follow the instructions in the *Help* menu to access the online manual, shown in Figure 4.3a. The setup parameters (classified into three parts: network, workload and algorithms) can be retrieved from the database as shown in Figure 4.3b, or specified by a corresponding sub-menu, *New Settings*. Some parameters depend on each other. For example, if "hot-spot" is specified as the communication pattern, a pair of parameters (x, y) should be defined afterwards. In this case, the GUI

will analyze the dependencies and pop up corresponding windows, until all necessary parameters are specified, as shown in Figure 4.3 c and d. Finally, the parameters specified can be saved into the database though use of the corresponding menu, just like the retrieval.

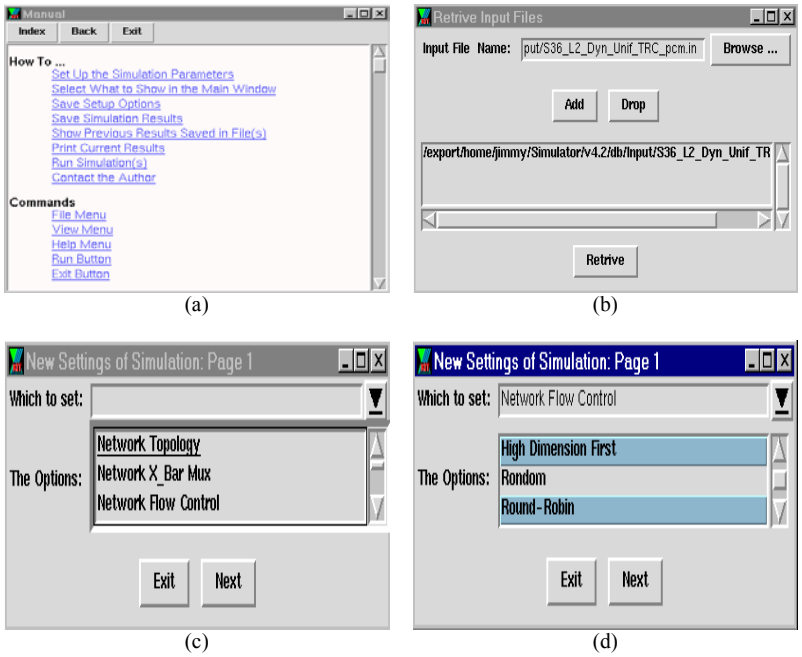


Figure 4.3. The online Manuel and Setup process of the simulator. (a) Manual. (b) Retrieve previous setup. (c) Choose the sub-domain to be explored. (d) Specify the design option(s).

After setup, the simulation is run by pushing the *Start* button; the results displayed can be stored into the database by pushing *Upload*. Previously saved results (in ASCII format) can be retrieved from the database and shown as the currently running results: the setup parameters and the graphic data are parsed from the original record.

The network is monitored with several parameters after starting the simulation. A packet is viewed as injected into the network as long as its header enters into the injector interface buffer of the attached router, and it is viewed as ejected as long as its tail leaves off the corresponding output ejector interface buffer of the attached router. The number of packets injected and ejected is checked globally to calculate the traffic rate, defined as:

Injected Traffic = Number of Packets Injected * Packet Size / Number of Nodes / Cycles passed,

Accepted Traffic = Number of Packets Ejected * Packet Size / Number of Nodes / Cycles passed.

The latency (L_i) for a packet (P_i) is defined as the number of elapsed cycles between the cycle (C_{Inj}) when it is injected and the cycle (C_{Ej}) when it is ejected,

$L_i = C_{Ej} - C_{Inj}$.

But only the average latency for all packets ejected (total number of N) is considered and defined as

Average Latency = $\sum_{i=0}^{N} L_i / N$.

For a dynamic workload, the traffic and the average latency can be shown in an often-used form as Figure 4.4.

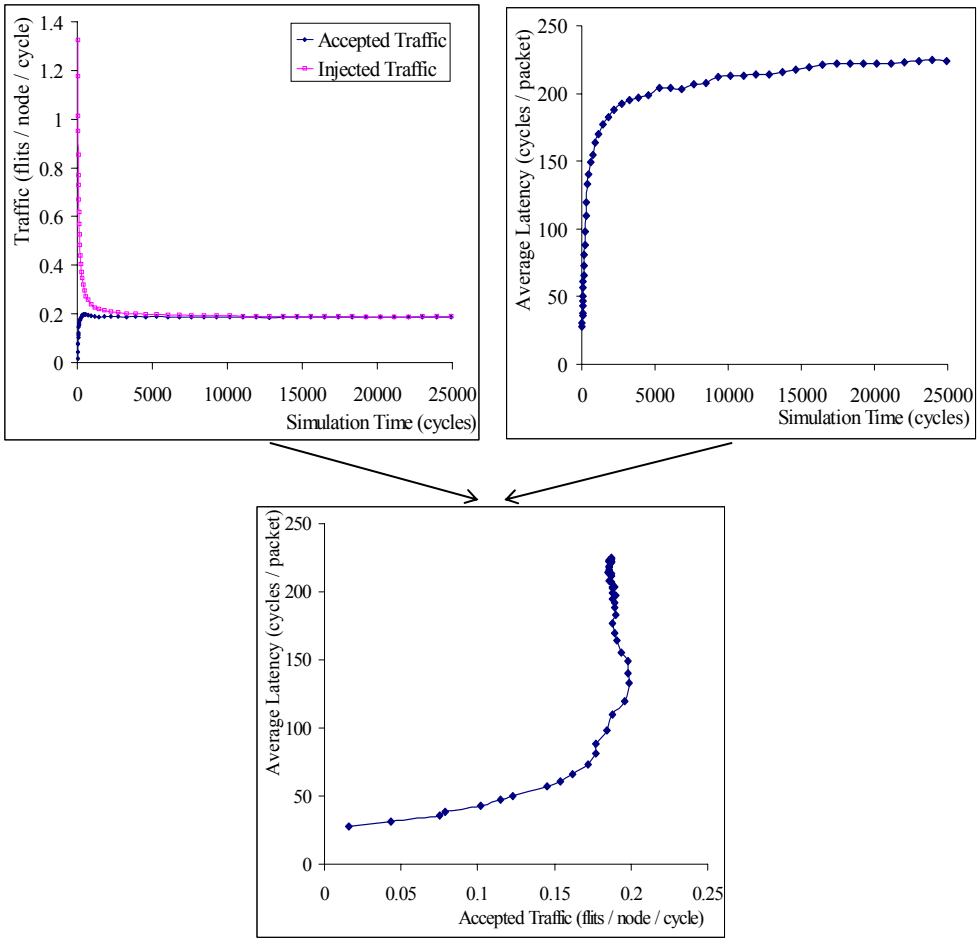


Figure 4.4. Monitor and sample the simulated network.

Figure 4.4 gives some clues on how to sample the running process. The ejection rate is pulled up following the decreasing of injection rate at the beginning, while average latency increases rapidly. This is because during this period most of the nodes are allowed to inject packets into the network while few packets have finished. This leads to the throttle of the injection and a stable situation, where the injection and ejection rates tend to equal and the average latency tends to a constant. It is not surprising, however, that at the very beginning the injection rate is bigger than one because of the way the injection rate is defined.

To grasp the overall model behavior, the performance is pictured at snapshot points instead of after each cycle. The model behavior shown implies a small snapshot-window during start up and a larger window near the saturation point. As maximum accepted traffic, or throughput, is a major concern, this guarantees more points observed during the stable state. The snapshot window used in the simulator is an exponential function of a ratio of the difference between the injected traffic and the accepted traffic over the injected traffic. For a static workload, the snapshot window is adjusted so that at each 10% of packets ejected, the related behavior is recorded.

If a static workload is applied, the simulation is terminated after all packets get to their destinations. If a dynamic workload is applied, there are two scenarios. The network may reach its saturation state, which is characterized by nearly the same injection and ejection rate, then the simulation should be terminated. The network may, however, decrease its ejection rate after reaching its maximum injection rate, especially when adaptive schemes are used. To differentiate the performance degradation from temporal fluctuation, a special window is used to monitor the relative difference between the injection and ejection rates for several points. If the performance degrades consecutively for each point in the window, the simulation is terminated.

By using this simulator, various design tradeoffs in the three categories, networks, workloads and algorithms, can be evaluated, with the emphasis on evaluating the performance of various routing algorithms under various network and workload parameters.

4.2 Network Design Tradeoffs

Various network design tradeoffs, and their effects on the relative performance of routing algorithms, are now explored.

4.2.1 Arbitration Policy

Arbitration takes place to resolve conflicting requests for shared physical links by several virtual channels and for the output virtual channel by packets residing in a few input virtual channels. To simplify the problem, three often-used policies are evaluated by using the TRC algorithm under a dynamic workload, with the throughputs shown in Table 4.1. The Fixed-Order policy gives priority to requests by the order the source is numbered. For example, the two virtual channels used to prevent deadlock in each direction are numbered 0 and 1, and the Highest-Number-First, is one implementation of this policy.

Table 4.1. The effect of arbitration policies on the throughput.

XBar Policy	Fixed Order	Fixed Order	Fixed Order	Random	Random	Random	Round -Robin	Round -Robin	Round -Robin
PC_Mux Policy	Random	Round -Robin	Fixed Order	Random	Round -Robin	Fixed Order	Random	Round -Robin	Fixed Order
Through-put	.192	.189	.182	.222	.225	.215	.234	.236	.226

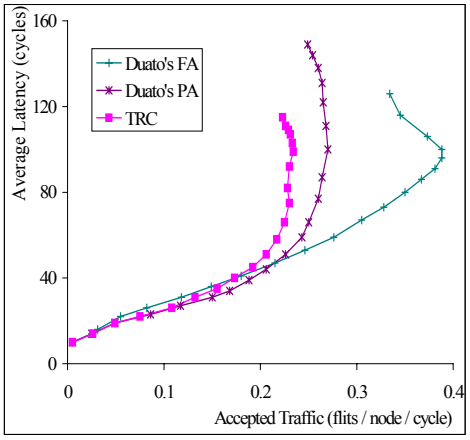
Note: Under environment: TRC routing, 16x16 Torus, 1 lane per VC, 104 flits total node buffer, 8-flit packet, uniform traffic)

The Fixed-Order arbitration performs the worst besides its potential of starvation. The random arbitration gets nearly the same performance as the round-robin, but generating the true random numbers by hardware is not as simple as generating pseudo random numbers. So, the round-robin policy is used afterwards in this dissertation.

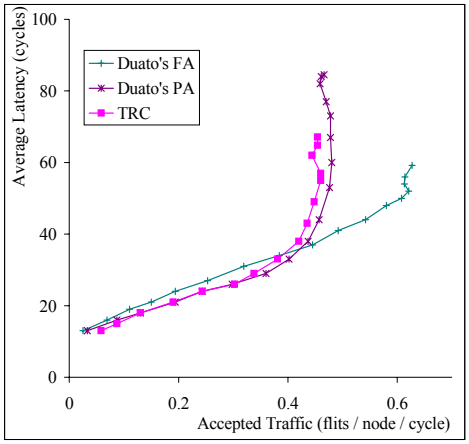
4.2.2 Network Size

Scalability is expected for any design, so a large network is preferred for evaluation purposes. But simulating a very big network may take too many computing resources and may become prohibitively time consuming for exploring a large design space. Since only relative performance is an issue, finding a representative size is sufficient to draw conclusions.

Figure 4.5a shows the performance of five algorithms on an 8x8 torus, which is pretty small. As expected, Duato's FA outperforms others, but the difference between Duato's PA and TRC is a little bit discouraging. Increasing the size to 16x16, the difference becomes more visible, as shown in Figure 4.5b. The reason is that the network bisection bandwidth (increased by two times) is competed for by more nodes



(increased by four times), and each packet has to travel a longer distance on average (increased by two times). The adaptive use of VCs in each direction benefits more in the 16x16 network than in the 8x8.



(a) (b)

Figure 4.5. The adaptivity is more likely beneficial to performance in large networks. (a). 8x8. (b). 16x16. (Under environment: total 9 lanes and a 108-flits node buffer for TRC, Duato's PA; 13 lanes and a 104-flits node buffer for Duato's FA. 8-flits packet, uniform traffic).

Further doubling the network width, as shown in Figure 4.6, does not affect the relative performance significantly. A common example, like a highway system, may be helpful to explain this phenomenon.

When it is lightly loaded, very few segments (if any) will become congested, therefore a complicated flow control is either not necessary or has little effect. When it contains moderate traffic, congestion may happen frequently, and the proper flow control may have a significant effect on the throughput. But if it is very heavily loaded, congestion may happen, continue, and spread everywhere. The flow control, no matter how clever it is, can not offer more bandwidth to the heavily loaded traffic, and throttling the incoming traffic becomes the final strategy.

Running simulations on a very large network is not only unnecessary to get the relative performance evaluated, but also out of the domain we assumed. Based on the large sets of tradeoffs explored, the investigation of the current commercial MPP machines and the near future scale of our proposed SoC multicomputer, the evaluations proceeded following are concentrated upon a medium sized network unless explicitly specified.

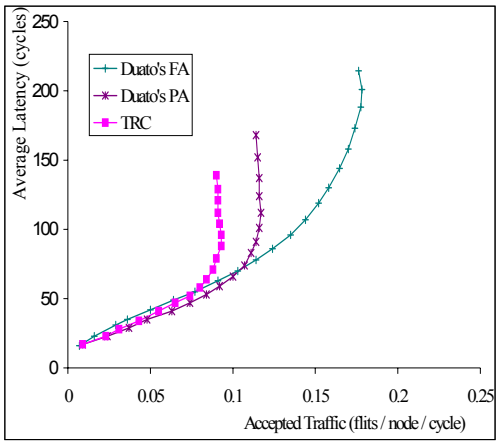


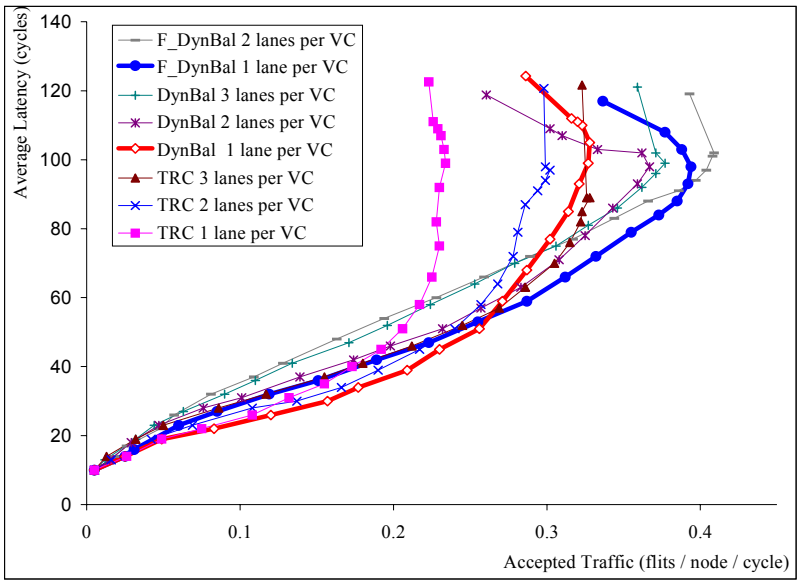
Figure 4.6. Relative performance on a 32x32 torus is similar to that on a 16x16 torus (Figure 4.5b). (Under environment: total 9 lanes and a 108-flit node buffer for TRC, Duato's PA; 13 lanes and a 104-flit node buffer for Duato's FA. 8-flits packet, uniform traffic).

4.2.3 Number of Lanes

It is found by previous research [5][26] that using more than four virtual channels is not cost-effective. If lanes are introduced into virtual channels, they can be treated the same way as virtual channels

for a specific algorithm. More lanes not only take more chip area by their corresponding buffers, but also increase the complexity of the internal switch, thereby slowing down the control cycle.

Figure 4.7 shows the performance of three algorithms with different lanes. What is especially interesting here is that the lanes, as one kind of router resource, are used with quite different efficiency. The



DynBal algorithm, with only one lane per VC (which is equivalent to two lanes per physical link), has nearly the same performance as TRC with three lanes per VC (which is equivalent to six lanes per physical link). The F_DynBal algorithm with one lane per VC, equivalent to three lanes per physical link, outperforms the DynBal with three lanes per VC, equivalent to six lanes per link. On the other hand, doubling the lanes per VC has decreased significance for F_DynBal than that for DynBal, and for DynBal than that for TRC. The reason is that the buffer space will be less than half and more links will be held by blocked packets, and that dynamic balance and adaptivity make an efficient use of lanes.

Figure 4.7. Dynamic balance and adaptivity leads to an efficient use of lanes. (Under environment: a 108-flit node buffer for TRC, DynBal, and a 104-flit node buffer for F_DynBal. 8-flits packet, uniform traffic).

4.2.4 Wiring Delay

All the previous presentations are based on the assumption that the physical wiring delay is limited to such a range that it will not introduce extra cycles for a packet transmitted from the output to the downstream input. This is true for moderate size systems constructed with tightly coupled components, either on chip, on board, or even within the same box. If the system becomes larger, and the router becomes faster following the trends of process technology [73], then wiring delay may become the critical part of the design. Figure 4.8 shows the performance for several algorithms under two different wire delays, assuming a pipelined transmission link [65].

Several discoveries can be derived from this picture:

- The bubble problem inherited in Duato's PA and Duato's FA leads to degraded performance when more wire delay cycles are considered. The basic requirement of the wire delay model is the large space required (2X rule) for the input buffers. Because of the non-packet-co-residence assumption of these algorithms, more bubbles will be introduced. Compared to Figure 4.5b, the benefit of adaptivity in these algorithms no longer exists.
- While the throughputs of T3D-like, DynBal and their associated fully adaptive routing algorithms are little affected by the wire delay, those of TRC, Duato's PA and Duato's FA are relatively much lower for longer delays. The former set of algorithms distributes the workload into lanes and buffers more evenly than that the latter set does. If a pipelined transmission link is used, the balanced use of buffers will lead to a more efficient use of links; otherwise the bubbles would hold the link.
- For any algorithm, the average latency is much higher for longer wire delays than that for shorter, even though the pipelined link is used. Pipelined links make efficient use of link capacity, but each flit has to pass the link for a prolonged time.

Unless specified, the evaluation in following sections will concentrate on the case that flits transferring from an output buffer to its downstream input buffer will take only one cycle.

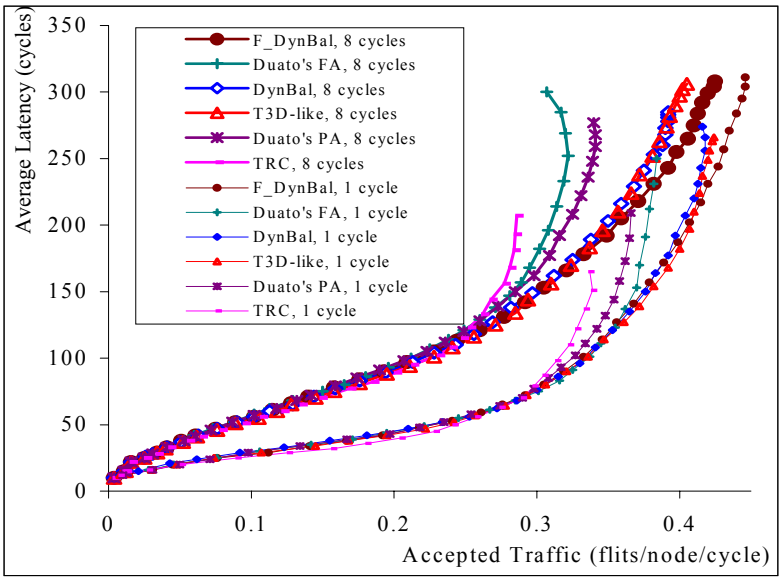
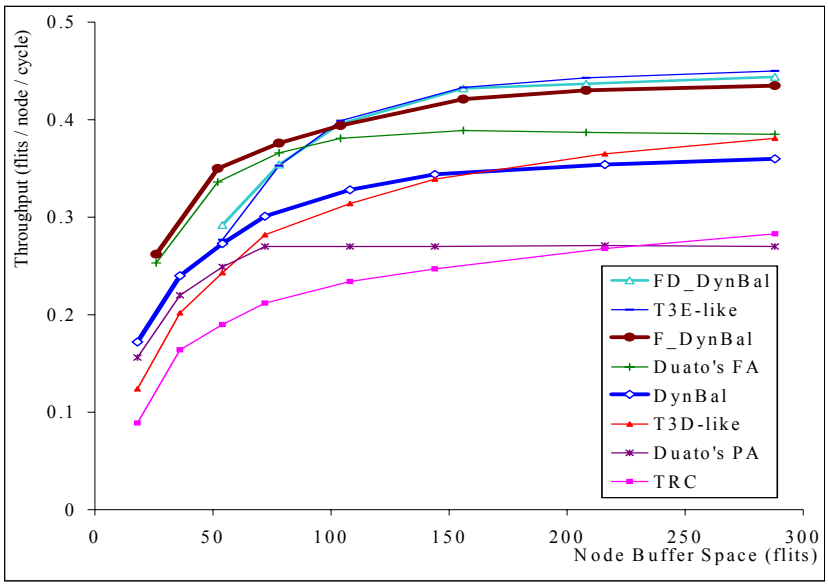


Figure 4.8. The effect of wire delay on routing performance. (Under environment: total 18 lanes and a 432-flit node buffer for TRC, T3D-like, DynBal, Duato's PA; total 13 lanes and a 442-flit node buffer for F_DynBal, Duato's FA. 8-flit packet, uniform traffic).

4.2.5 Node Buffer Space

For a particular algorithm and packet size, the buffer size affects performance in several ways. A minimum of two flits per buffer is required for asynchronous PC wormhole routing; otherwise it would introduce bubbles [4][26][30]. When the buffer size gets bigger, blocked packets hold fewer buffers as well as fewer links so that a big buffer improves the performance. However, increasing buffer size also



exacerbates the HOL (Head-Of-Line) problem, so there is an optimal size that should be considered.

Figure 4.9. The effect of buffer space to various routing algorithms. (Under environment: total 9 lanes for TRC, T3D-like, DynBal, and Duato's PA; 13 lanes for all others. 8-flits-packet, uniform traffic).

Various algorithms use buffer space with different efficiency, along with what we have mentioned for the use of lanes and links. As the drain-off-buffer based algorithms rely on buffer space, they are also put together with wormhole routing algorithms in Figure 4.9. If the buffer is small, then the performance of non-fully adaptive routing algorithms, using the same number of lanes, is ordered as DynBal, Duato's PA, T3D-like and TRC. Both of the two fully adaptive algorithms perform better than non-fully adaptive ones. If the buffer gets bigger, the bubble problem of Duato's PA and Duato's FA becomes significant. If the buffer is so big that drain-off buffer based algorithms can be applied, drain-off based routing outperforms other pure wormhole routing, and the static balance performs a little bit better than dynamic balance due to the HOL problem in the latter. In general, the dynamic balance scheme is more flexible and more efficient in use of buffer space, and this becomes more obvious in a variable packet-sized system.

4.3 Workload

In this section, routing algorithms are compared by using various workloads.

4.3.1 Packet Size

Contrasting with Figure 4.9, Figure 4.10 shows the performance of algorithms under the nearly same buffer space but with varying packet size [44].

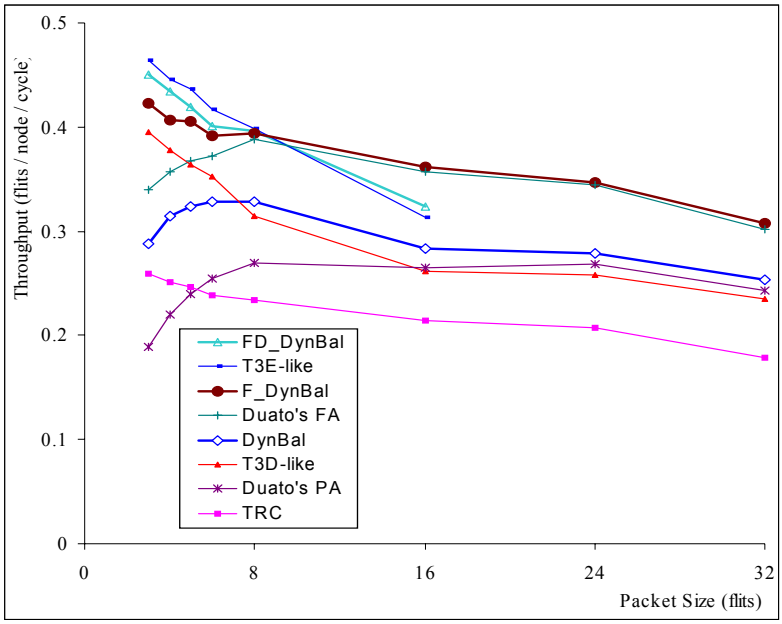


Figure 4.10. The effect of packet size to various routing algorithms. (Under environment: total 9 lanes and a108-flit node buffer for TRC, T3D-like, DynBal, Duato's PA and VCT; 13 lanes and a 104-flit node buffer for all others. uniform traffic).

The issues of buffer space and packet size are actually tightly coupled, rather than independent of each other. It is the ratio of lane buffer size over packet size that affects performance. If the ratio is greater than one, a packet can be held in one buffer, which is the basic requirement of VCT and drain-off buffer based hybrid algorithms. The reciprocal of the ratio is related to the number of links held when a packet is blocked, so it reflects an aspect of the network congestion. If the maximum packet size is pre-defined, and the lane buffer size is big enough to hold it, F_DynBal performs better than non-fully adaptive schemes, but worse than the drain-off buffer based algorithms. The Duato's PA and Duato's FA still have the bubble problem, as discussed in section 4.2.5. If the packet gets bigger, but the drain-off buffer still can hold multiple packets, then the drain-off based algorithms perform worse than the pure fully adaptive wormhole routing algorithms. It is clear that the better-performance range of DynBal and F_DynBal algorithms is

broader than that of others in the set of non-fully adaptive and fully adaptive routing, especially in a variable-length-packet system.

4.3.2 Local Traffic

Consider the routing performance, shown in Figure 4.11, under an array operation such as

$$A(I,J) = A(I+k,J+k).$$

The packet in a node should be transferred to a node diagonally k-hops away. Although the operation is executed globally, the message passing is restricted in a local manner.

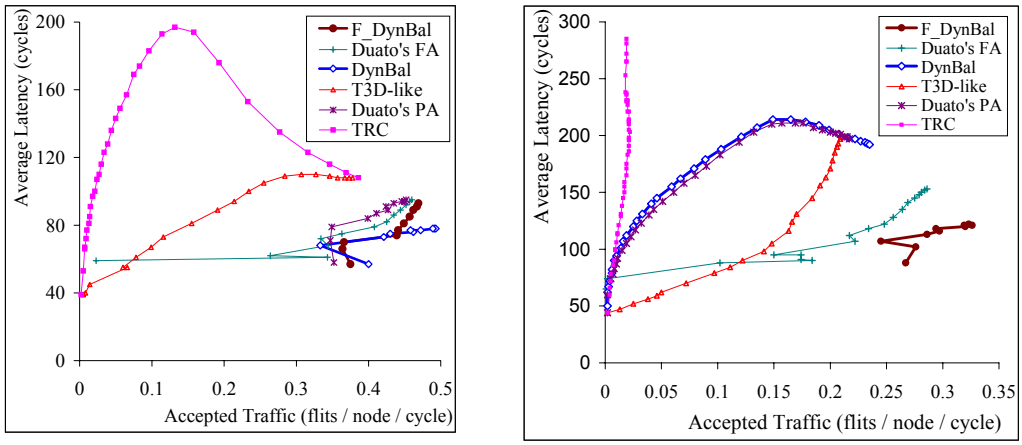


Figure 4.11. The effect of a local workload. (a) k = 2. (b) k = 3. (Under environment: total 9 lanes and a 108-flit node buffer for TRC, T3D-like, DynBal, Duato's PA; 13 lanes and a 104-flit node buffer for all others. 24-flits packet).

There are two questions that need to be answered before comparing the relative performance of the algorithms. The first is why the average latency goes up and down for some of the algorithms rather than going higher and higher as approaching the saturation point. To answer this question, using TRC as an example, the traffic and average latency are monitored as in Figure 4.12.

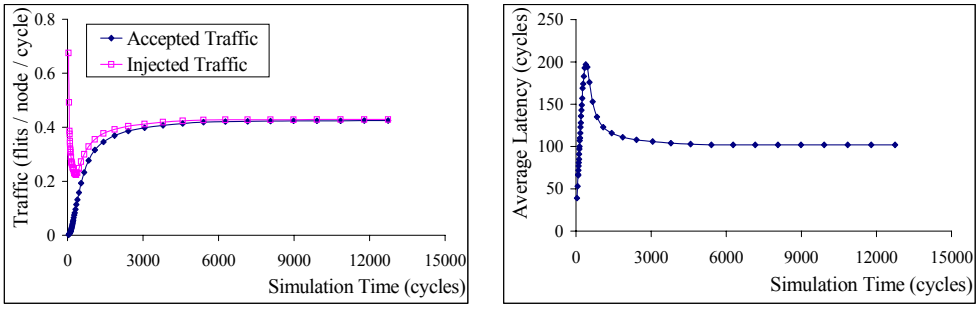


Figure 4.12. The injection throttle of TRC under local traffic.

Two major procedures are involved:

- As a normal behavior (see Figure 4.4), when started up, more packets can be injected and few ejected, the injection rate goes down and ejection rate goes up after a few more packets are injected. More packets in the network cause the average latency to go up sharply.
- Because the traffic is localized, packets can finish in just a few hops, which allows an increased ejection rate to continue and leaves more space in the injection buffer, so that the injection rate goes up afterwards. The automatic injection throttling adjusts the network congestion status. The network runs in a stable state where both rates are balanced, and packets proceed to their nearby destinations with nearly constant time.

In Figure 4.11a, TRC undergoes a typical case of this procedure because most of the packets are injected into the same virtual channel. T3D-like also performs similarly because the globally optimization on virtual channel assignments does not prevent unbalanced use of them locally. Then, why they do not behave in this way in Figure 4.11b? Keep in mind that both the local unbalance and the throttled injection are the co-creators of this phenomenon. If packets take longer routes to their destinations, the potential for more congestion cannot be attributed to promptly throttling injection. Therefore the network will be quickly saturated with much higher average latency. Instead, the DynBal and Duato's PA algorithms in Figure 4.11b perform similar to T3D-like in Figure 4.11a. This is because of the limited adaptivity, although is not enough to increase the throughput; it can react more quickly than TRC and T3D-like to feedback of the status and cause the throttle.

Another question is why the start point of the accepted traffic for some algorithms is significantly later than that for other algorithms and what causes the fluctuation of the accepted traffic. This is due to the modeling as mentioned in section 4.1.4. The network status is sampled and recorded only when there is at least one packet finished. Some algorithms, for example, the F_DynBal allow packets to be injected to more virtual channels than others do, the TRC, for an example. If the load is highly local, then the probability that more packets can be ejected at the same time is very high, which means a much higher accepted traffic during the 'warm-up' period. The monitored traffic fluctuation is caused by the unified

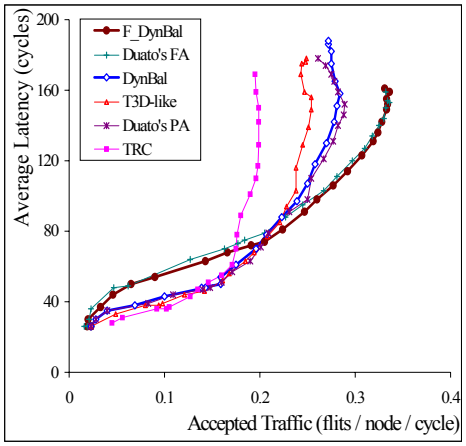
snapshot window size for various traffic loads, which emphasize the stable state with a longer window, and use a smaller window for the 'warm-up' period. A larger window size may be helpful for this specific load, but may result in too many points recorded for other loads.

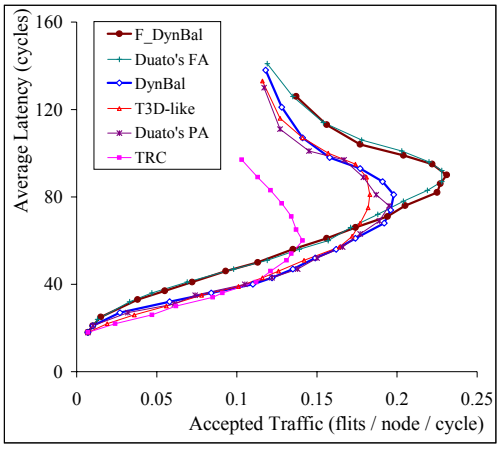
Thus, several observations may be drawn here:

- Even though T3D-like may perform a little bit better than DynBal, in the case of a relative bigger ratio of lane buffer size to packet size under uniform traffic, the globally and statically optimized virtual channel assignments do not result in better performance under a local traffic.
- DynBal is the most effective algorithm for highly local loads. Fully adaptive algorithms perform nearly the same as DynBal with higher cost (the number of lanes and potentially lower speed).
- Fully adaptive routing is not as useful for highly local loads as for other traffic. The merits of fully adaptive algorithms become significant when locality increases.

4.3.3 Non-uniform Traffic

Figure 4.13 shows the effect of two kinds of non-uniform traffic, hot-spot and random bit vector, on pure wormhole algorithms. Both of these two workloads have some degree of randomness. The DynBal algorithm still outperforms the T3D-like similarly as for local traffic, but the difference between DynBal and Duato's PA, as well as the difference between F_DynBal and Duato's FA, becomes smaller than for uniform workloads. This implies that the merits gained in each single ring, such as removing bottlenecks and bubbles, are not enough to lead to better performance under the traffic not uniformly distributed among rings, which may need more adaptivity on turns. On the other hand, partial adaptivity and the balanced use of virtual channels do play a part on performance improvement over pure deterministic routing (TRC).





(a) Hot-Spots (4,16) (b) Random Bit Vector

Figure 4.13. The effect of non-uniform traffic to wormhole routing algorithms. (a) Hot-Spots (4,16). (b) Random Bit Vector. (Under environment: total 9 lanes and a 108-flit node buffer for TRC, T3D-like, DynBal, Duato's PA; 13 lanes and a 104-flit node buffer for all others. 16-flit packet for Hot-Spots, 24-flit packet for Random Bit Vector traffic).

The adaptivity in choosing virtual channels within a single ring may not be enough to enhance performance under other non-uniform traffic loads. Let's take Bit-Reversal as an example. Consider VCT and drain-off buffer based algorithms as shown in Figure 4.14a for a larger buffer space than in Figure 4.13. With the same design parameters, the performance under uniform traffic is illustrated in Figure 4.14b. The non-fully adaptive routing algorithms make little difference with each other for Bit-Reversal traffic while the partial adaptivity and balanced virtual assignments improve performance over TRC as much as 40% for uniform traffic. The fully adaptive wormhole routing algorithms perform nearly the same as hybrid

algorithms, and outperform TRC by 77% for uniform traffic. Under Bit-Reversal, they outperform TRC by 84%, but under-perform the hybrid schemes, which have more than double the throughput of TRC. The performance gain of hybrid schemes over fully adaptive wormhole routing algorithms, more notable in Figure 4.14a than in Figure 4.14b, is based on the assumption that the maximum packet size is pre-defined and the lane buffer is big enough to hold at least one packet.

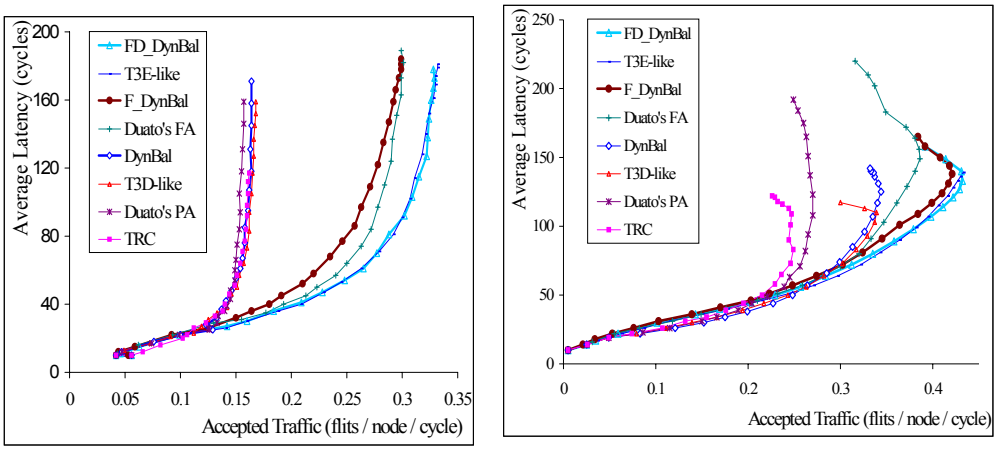


Figure 4.14. The effect of Bit-Reversal (a) and uniform traffic (b) to WH and hybrid of VCT and WH routing algorithms. (Under environment: total 9 lanes and a 144-flit node buffer for TRC, T3D-like, DynBal, Duato's PA; 13 lanes and a 156-flit node buffer for all others. 8-flit packet).

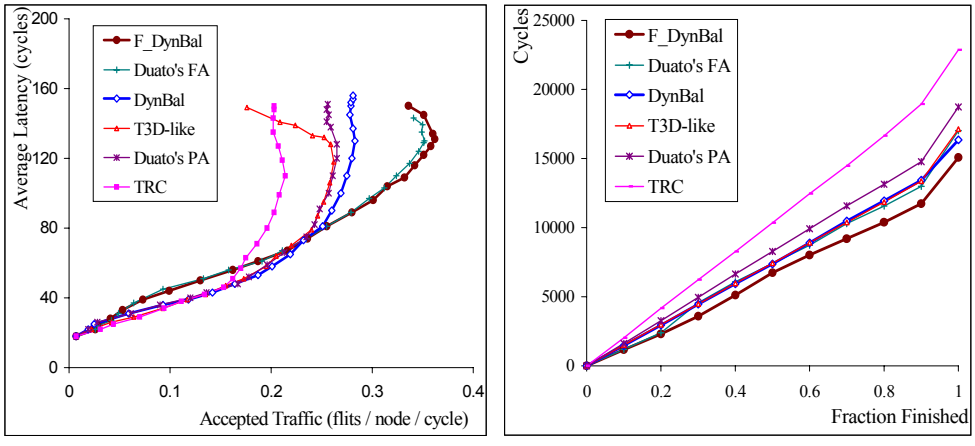


Figure 4.15. Wormhole routing under dynamic (a) and static (b) workloads. (Under environment: total 9 lanes and a 108-flit node buffer for TRC, T3D-like, DynBal, Duato's PA, 13 lanes and a 104-flit node buffer for all others, 16-flit packet).

4.3.4 Static Workload

Static workloads are applied to measure the time (in cycles here) that is necessary to complete a task, while dynamic workload is used to measure the throughput. As shown in Figure 4.15, there are some discrepancies between the performances under static and dynamic workload. For example, the Duato's PA and T3D-like achieve the same throughput, but T3D-like needs fewer cycles to get the job done.

The performance discrepancies can be explained by the different use of network capacity during the static workload delivery process, as shown in Figure 4.16. Most of the time, T3D-like can use higher network capacity than Duato's PA for the transmission. The intensive bubbles introduced by Duato's PA, especially when more buffers have already resided some flits of other packets, cause inefficient use of bandwidth. This phenomenon is more typical for Duato's FA. Both Duato's FA and F_DynBal reach nearly the same delivery rate after 10% of the packets being drawn, but the rate is forced to slow down more quickly for Duato's FA because more 'cyclic-channel' buffers need to be emptied before accepting other packets. This leads to the better performance of F_DynBal over Duato's FA under static workload than in dynamic workload.

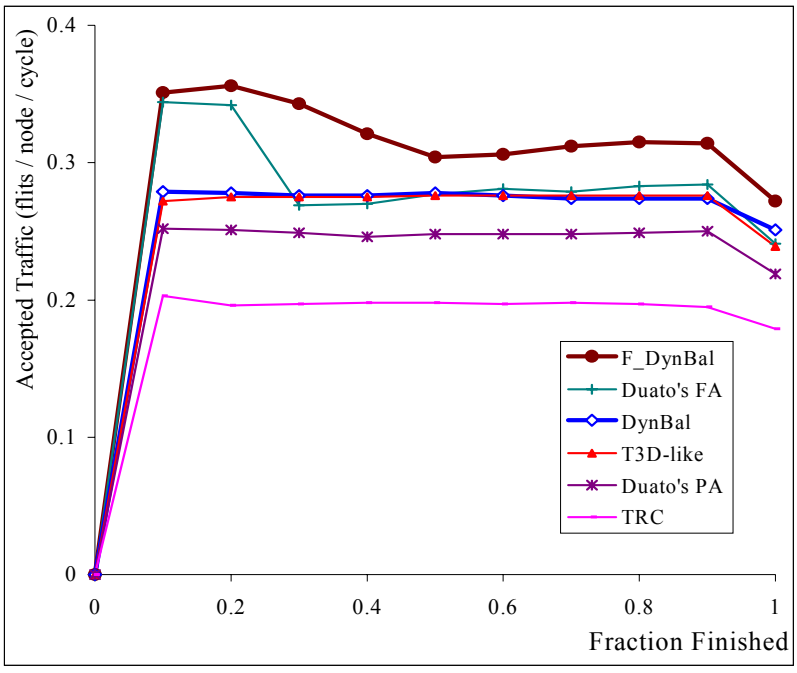


Figure 4.16. The throughput utilization during the distribution of a static workload. (Under environment: total 9 lanes and a 108-flit node buffer for TRC, T3D-like, DynBal, Duato's PA; 13 lanes and a 104-flit node buffer for all others. 16-flit packet).

4.4 Routing Performance Summary

More evaluations on the performance of routing algorithms have been made and combined with the results discussed above. These are shown in Table 4.2 and Table 4.3, for a small and a big ratio of lane buffer size to packet size respectively.

Table 4.2. The throughputs of the network with a small ratio of buffer size over packet size.

	Uniform	Random Near2	Random Near4	Random Near8	Diagnal Shift 2	Diagnal Shift 3	Dimension Reversal	Bit Reversal	Random Bit Vector	Hot Spots
TRC	.214	.548	.340	.191	.408	.024	.186	.160	.200	.141
T3D-like	.262	.587	.446	.281	.426	.221	.181	.167	.267	.183
DynBal	**.283**	**.645**	**.464**	**.281**	**.492**	**.225**	**.182**	**.167**	**.297**	**.198**
Duato's PA	.265	.626	.442	.259	.460	.224	.181	.164	.283	.195
F_DynBal	**.362**	**.642**	**.512**	**.351**	**.477**	**.329**	**.276**	**.295**	**.349**	**.231**
Duato's FA	.357	.632	.503	.346	.466	.289	.279	.291	.358	.228
T3E-like	.313	.609	.471	.318	.430	.254	.262	.280	.333	.218
FD_DynBal	**.324**	**.603**	**.458**	**.316**	**.456**	**.284**	**.258**	**.282**	**.330**	**.215**

Note: Under environment: total 9 lanes and a 108-flit node buffer for TRC, T3D-like, DynBal, Duato's PA; 13 lanes and a 104-flit node buffer for all others. 16-flit packet).??????????????????????/

Table 4.3. The throughputs of the network with a big ratio of buffer size over packet size

	Uniform	Random Near2	Random Near4	Random Near8	Diagnal Shift 2	Diagnal Shift 3	Dimension Reversal	Bit Reversal	Random Bit Vector	Hot Spots
TRC	.247	.642	.421	.240	.491	.059	.186	.162	.239	.166
T3D-like	.339	.697	.556	.359	.491	.236	.181	.168	.329	.227
DynBal	**.344**	**.760**	**.552**	**.338**	**.419**	**.203**	**.181**	**.164**	**.336**	**.214**
Duato's PA	.270	.686	.466	.263	.416	.180	.172	.157	.297	.185
F_DynBal	**.421**	**.792**	**.589**	**.403**	**.460**	**.303**	**.279**	**.299**	**.394**	**.248**

Duato's FA	.389	.747	.556	.383	.450	.278	.291	.301	.379	.244
T3E-like	.433	.793	.600	.409	.453	.303	,307	.332	.406	.245
FD_DynBal	**.432**	**.798**	**.589**	**.402**	**.465**	**.305**	**.308**	**.329**	**.398**	**.241**

Note: Under environment: total 9 lanes and a 144-flit node buffer for TRC, T3D-like, DynBal, Duato's PA; 13 lanes and a 156-flit node buffer for all others. 8-flits packet). ??????????????????

Based on the experiments conducted, several conclusions can be derived on the relative performance of routing algorithms under other design tradeoffs:

- Compared to static optimization of virtual channel assignments, dynamic balance performs better with lower estimated cost. The DynBal outperforms TRC by up to 40%. It can perform better than T3D-like by up to 25% under local traffic. F_DynBal outperforms T3E-like and T3D-like by up to 15% and 38% respectively in the case of large packets.
- Compared to partial adaptive selection of virtual channels, dynamic balance performs better by removing bottlenecks and reducing bubbles. DynBal may outperform Duato's PA by up to 27% and F_DynBal can be up to 8% better than Duato's FA, in the case of a large ratio of buffer size to packet size.
- Fully adaptive routing performs better under more workloads and may be more cost-effective with respect to estimated cost of buffers (lanes). F_DynBal outperforms TRC and DynBal by 158% and 69%, respectively, under the perfect shuffle.
- For a system in which the maximal packet size is predefined and the buffer space is sufficiently large, drain-off buffer based routing (a hybrid of VCT and WH) may or may not outperform a general fully adaptive WH routing, depending on the relative size of the buffer and the packet.

Hence, the fully adaptive buffered WH routing with dynamic virtual channel balance (F_DynBal) should be viewed as a more flexible scheme that performs best in the domain of variable packet-size systems; that is, those that do not guarantee buffer space for a maximum packet.

5 ASIC DESIGN OF ADAPTIVE ROUTERS

A fully adaptive wormhole router that implements the F_DynBal algorithm is designed and synthesized onto the LSI-G11-p ASIC technology. The implementation cost in chip area and the timing are then extracted and combined with the performance simulation results of the previous chapter to derive cost-effectiveness evaluations.

5.1 Overall Router Model

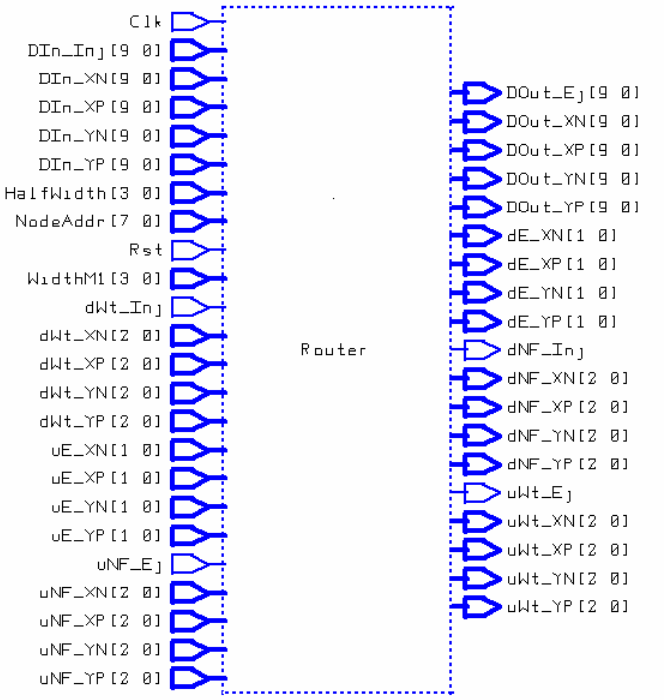
The canonical model is transferred to a schematic representation. The tradeoffs on the architecture issues and design procedures are briefly discussed.

5.1.1 Schematic Model

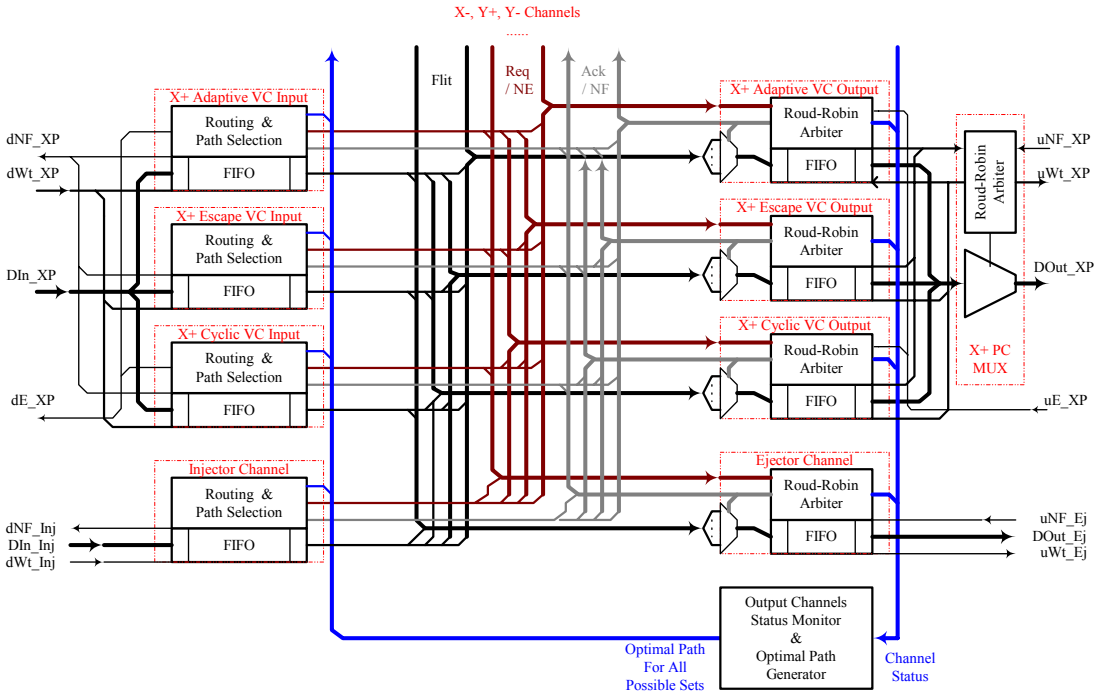
The model is illustrated in Figure 5.1. It is based on the canonical model developed in Section 4.1.1. We assume routing by the F_DynBal algorithm on a two dimensional torus with a physical link for each direction. For clarity, only part of the router, the schematic graph on the positive direction along x dimension and the local node interface (injection and ejection) is presented in Figure 5.1b.

There are two ways that a flit can be transmitted (Wt) into the input buffer if it has space (NF): either from a local node injector or from an up-stream node output buffer via the physical channel multiplexer (PC MUX). The header flit of a packet is decoded for all of the possible paths it can take, while an optimal path for each virtual channel is generated based on the status of all related output channels. The optimal path is requested (Req) and may be granted (Ack) by a round-robin arbiter sitting on each output virtual channel. A packet will hold the granted path to guarantee its integrity. Flits in the input buffer, if any (NE), can then be transmitted into the output buffer if space is available (NF). Virtual channels (or lanes) in the same direction share a physical channel at a flit-level and are allocated in round-robin order. The packet

can then be transmitted to a local ejector, or to the down-stream-input buffer via the multiplexed physical link.



(a)



(b)

Figure 5.1. The logic representation and schematic graph of the F_DynBal router. (a) Logic representation. (b) Schematic graph of the XP channel and the interface to the local injector/ejector.

5.1.2 Design Procedures

Three basic steps are involved in the design of both the overall router and its parts: modeling, logic verification, and synthesis. The design is first modeled [43]; the general conclusions may be represented in schematic graphs. The model is then described in Verilog HDL [58] and the logic can then be verified by using simulation tools like SILOS. Finally, the design can be mapped into a specific process technology, LSI G11-p [49][50] here, to be synthesized and optimized for area cost and clock frequency [8][9][15][75~77].

The schematic model presented here is a result of several top-down and bottom-up procedures. The design is partitioned into several blocks (top-down), and each block can be designed following the three steps. A critical path is used to locate the bottleneck parts, which can then be modified for improvement. The overall design should then be re-verified and re-synthesized (bottom-up). The design can be optimized either by re-partition, or by parallelizing operations among, or within, blocks.

The synthesis tool used is Synopsys, which optimizes a design constrained either by timing or chip area [77]. The same design can achieve a higher clock frequency by using more space after optimization. When comparing different design strategies of the same logic implementation, timing is used as the primary target in this research.

5.1.3 Architectural Features

The architectural issues involved in the router design are listed bellow:

- Switch. The switch is used as the data path from input lanes to output lanes and is implemented as a multiplexer on the side of output channels. It is not implemented as a single crossbar module because, otherwise, extra cycles will be introduced.
- Data link width. The number of physical link bits (phits) is assumed equal to that of flow control bits (flits). The header flit of a packet contains eight bits for the destination node address (x, y)

and two bits for the header/tail tag. The buffer depth unit is a bit, and the buffer width is the same as a flit.

- Centralized output channel status monitor and optimal path generator. These facilitate the parallelization of header decoding and path selection.

5.2 Buffer Design

The input and output buffers are implemented as register FIFOs [58]. To facilitate the transmission control of the overall router design, the buffer should have the following features:

- NF signal, which is interpreted by the down-stream buffer control logic as space available in this buffer.
- NE signal, which is interpreted by the up-stream buffer control as there being at least one flit in this buffer. Signals from the up-stream virtual channel, both the local output buffer and its up-stream input channel, are combined and then used as a condition for the cyclic channel routing.
- Last signal, which is used by the associated flow control logic. When it is asserted and the FIFO is currently being read, no read signal can be issued to this buffer during the next cycle.
- Always driven output, is the basic requirement if the targeted upstream buffer wants to get the flit in the same cycle when the corresponding read signal is asserted.

5.2.1 Basic Design Tradeoffs

The FIFO can be implemented either as a circular queue or as a shift register. The circular queue implementation [58] has centralized control logic to select a buffer to write into or read from. Figure 5.2a shows a design where all buffer outputs are multiplexed into a single FIFO output. If the FIFO gets deeper, the large MUX not only takes much more chip area, but also adds more gate delays. Several tri-state drivers can replace the MUX, as shown in Figure 5.2b, which may reduce the space requirement and speed up the read operation [43]. But unfortunately, the ASIC library we used does not contain tri-state cells.

The control logic is distributed into each basic buffer when the FIFO is implemented as a shift register, as shown in Figure 5.2c. Read is always from the head of the FIFO, and write is always to the tail.

More status information, such as the validity of data in the last and the next slot, is needed for control. The advantage of this design is that the FIFO depth has little effect on its speed.

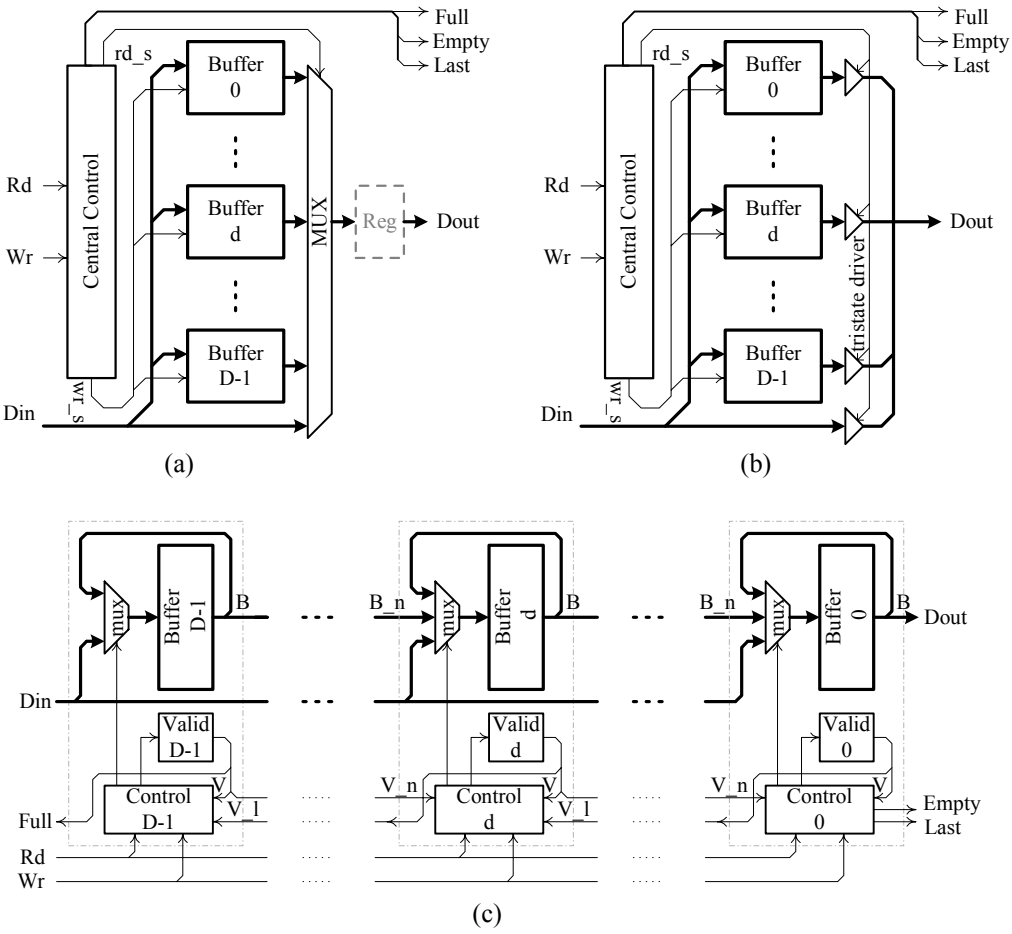


Figure 5.2. Basic FIFO design tradeoffs. (a) Circular queue implementation with big MUX. (b) Circular queue implementation with tri-state driver. (c) Shift registers implementation.

Table 5.1 shows the synthesis results of the two designs corresponding to Figures 5.2(a) and (c) for two FIFOs with fixed width (10 bits). The depth does not affect the speed of the latter, while the speed of the former becomes slower when the FIFO gets deeper. Because very deep FIFOs will also take too many registers, combining register with RAM, as used in [56], is a more realistic way than using register only.

Table 5.1. The cost and speed of the FIFO designed as a circular queue and as a shift register.

Depth (slots)	Area (cells)		Timing (ns)	
	Circular	Shift	Circular	Shift
4	4,189	4,460	.53	.65
16	18,092	22,991	.68	.65

5.2.2 Large FIFO

When a FIFO gets large, very wide and/or deep, cost and/or speed issues become significant. A control signal needs to drive each of the registers at the same depth. When the FIFO gets wider, more drivers are inserted to distribute the signal, like a tree as shown in Figure 5.3a, because both the fan-outs limitation and the timing constraint (more fan-outs corresponding to a slower gate). Each stage of the tree introduces extra gate delays so that a wide FIFO may be slower than a thinner one.

(a) (b)

Figure 5.3. Design strategy for a wide FIFO. (a) Driver tree. (b) Parallel control.

To speed up the FIFO operation, we can virtually split a bank of the FIFO into several slots and control them in parallel, as shown in Figure 5.3b. The number of tree stages, and hence the number of driver gate delays, is reduced. The choice of the number of slots depends on the technology used and the timing constraints expected -- unwisely chopping the FIFO into slots may increase space significantly without any speed gain. This is shown in Table 5.2 for a FIFO 32-bit wide and 4-bit deep using shift register.

Table 5.2. The cost and speed of FIFO designed by using parallel control.

Number of slots	Slot Width (bits)	Area (cells)	Timing (ns)
1	32	8,906	.70
8	4	18,262	.53

As mentioned earlier, to save the expensive registers required in a deep (maybe also wide) FIFO, combing the register and RAM in a FIFO design [56] is a good choice. The number of registers depends on memory access speed, while the overall speed predominately depends on the FIFO width.

5.3 Routing and Path Selection

Routing will generate all possible switch connections taken by the packet in the input channel, and the final path is decided based on the status of output channels and connections. Although the options for different packets may be different, the routing circuit sitting on the input channel must process all these probabilities, as shown in Table 5.3 by using F_DynBal.

Pkt Max Freedom / Channel Options	Input VC	Output VC Request List												
		Y+a	Y+e	Y+c	Y-a	Y-e	Y-c	X+a	X+e	X+c	X-a	X-e	X-c	Ej
4 / 10	Y+a	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓
3 / 4	Y+e	✓	✓	✓										✓
3 / 4	Y+c	✓	✓	✓										✓
4 / 10	Y-a				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3 / 4	Y-e				✓	✓	✓							✓
3 / 4	Y-c				✓	✓	✓							✓
4 / 10	X+a	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓
3 / 10	X+e	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓
3 / 10	X+c	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓
4 / 10	X-a	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓
3 / 10	X-e	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓
3 / 10	X-c	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓
3 / 13	Inj	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Max # of Requests		10	10	10	10	10	10	6	6	6	6	6	6	13

Table 5.3. Possible routing requests of each input channel by using F_DynBal.

The maximum freedom is the maximal number of paths that can be used for a specific packet in the input buffer, while the number of options for any packet in the input buffer is the number of channel options, which directly affects the routing complexity. All possible requests to a certain output channel are counted as the maximum requests, which contribute directly to the complexity of the arbitrator. The routing on the injector interface and the arbitration on the ejector interface are the most complicated ones in their set. Using the cost-speed model, it can be seen that parallel header decoding and status collection for path optimization can enhance performance with little cost.

5.3.1 General Approaches

From what has been analyzed in section 2.3.3, the two basic approaches [16][30] both implied a model shown in Figure 5.4, in which header decoding and path selection are done serially.

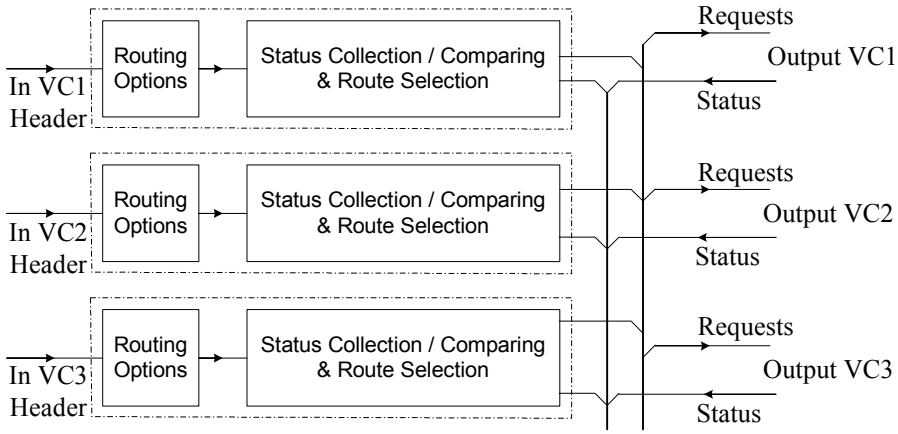


Figure 5.4. Serialized header decoding and path selecting approach.

Assuming a 0.8 micron gate array technology, the address decoding would take 2.70 ns, while the followed path selection function would spend 0.6 + 0.6 log4 = 1.8 ns. The time spent on these two parts dominates the speed of the router, especially when adaptive algorithms are used [26].

5.3.2 Parallel Address Decode and Status Comparison

Logically, nothing prevents the parallelization of decode and status comparison because the status has been stabilized during the decoding. Since much of the path selection functionality is to compare the gathered information from output channels, parallelism may dramatically speed up the routing process and thereby remove the critical path formed by complicated adaptive routing algorithms [16][26]. The idea is illustrated in Figure 5.5. A central status collection and comparison circuit generates the optimal path from each possible set. Then the actual routing optional set (of paths) generated after the header decoding is used to select the proper optimized path.

What is left, thereafter, are that the logic ‘glueing’ these two pieces, the multiplexer for the selection, must be fast, and that the overall changes should not introduce much overhead in chip area. The first can be

easily satisfied because the selection is just a part of the original path selection function. If the selection multiplexer is viewed as a part of the original, then the optimal selection in the central comparison module is the only change. Because the status comparison is centralized rather than sitting on each input channel as done originally, it will not introduce any overhead after duplicated computation circuits are removed.

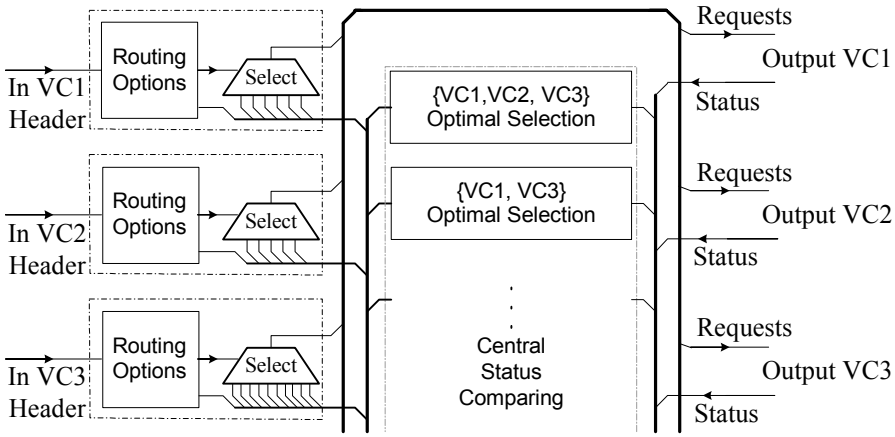


Figure 5.5. Parallel header decoding and status comparing.

5.4 Input Channel Design

Two major components, the input buffer and the routing control logic, constitute the input channel. Based on the critical path analysis of a simple design, parallelism is again introduced to improve the design.

5.4.1 Critical Path Analysis of a Simple Design

A simple design is schematically illustrated in Figure 5.6. The header is detected at the head of the input buffer. The destination address is compared with the local node address by routing logic to generate the possible routes. The problem here is that the FIFO operation and the routing together form the critical path and result in a slowdown of the circuit.

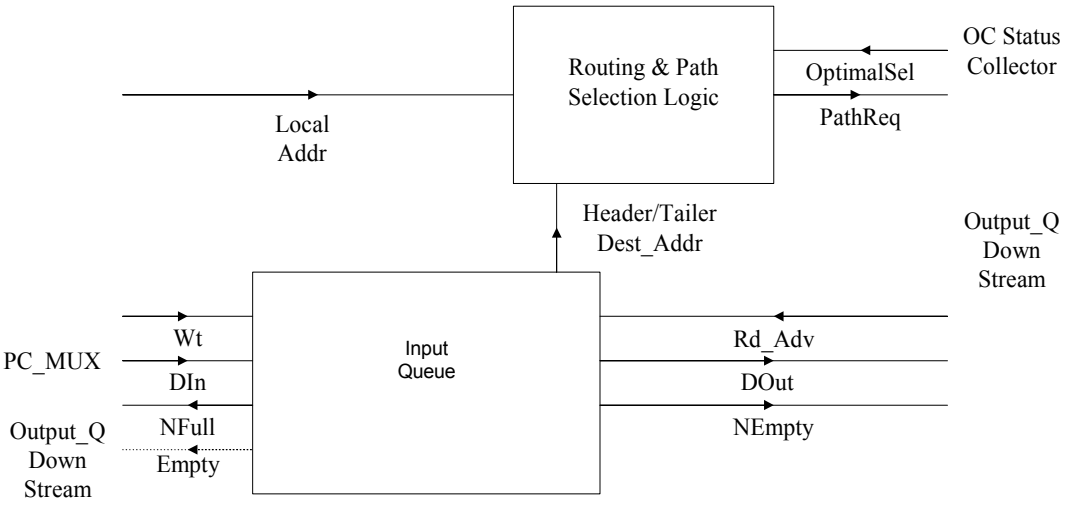


Figure 5.6. The schematic graph of a simple input channel design.

5.4.2 Parallel Routing and FIFO Operation

By parallelizing the routing and the FIFO operation, as shown schematically in Figure 5.7, the problem in the simple design can be resolved. Header detection is shifted to the trailing physical links and while the header fills the FIFO, its address is decoded and all optional paths are generated. If there are any flits of other packets ahead of this header in the FIFO, the optimal path will be updated at each cycle until this header appears at the head of the FIFO. Thereafter the most current optimal path would be requested. Because the FIFO output is always driven, the assertion of the grant signal (if no contention, it would be ready after being requested) would let the header flit directly fill in the granted output buffer.

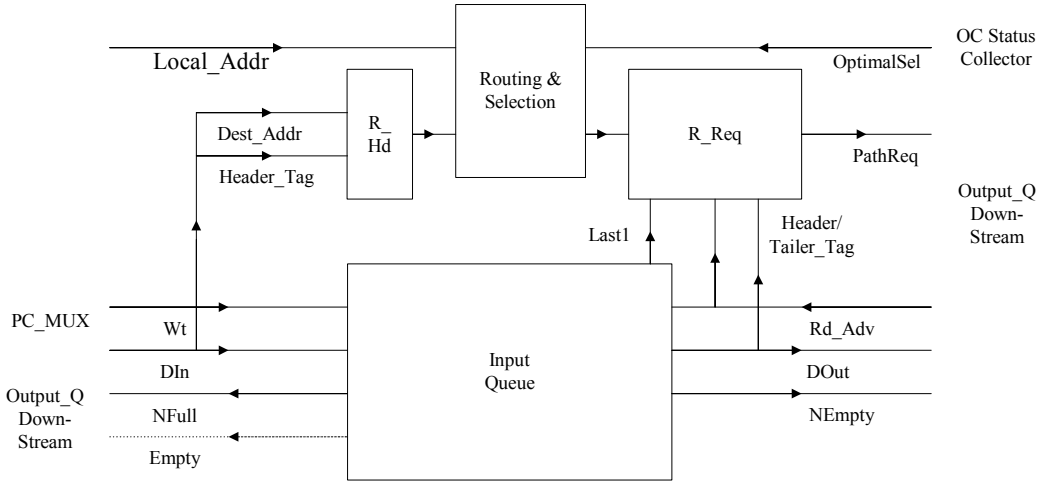


Figure 5.7. Input channel design by parallel routing with FIFO operation.

An implicit assumption made here is that the input buffer is smaller than the smallest packet in the system. In this case the circuit is simple. Otherwise, the path options generated for the header flits not taking the first place of the input buffer should be queued. The optimal path is effective only when its corresponding packet header appears at the head of the FIFO, and the slot for these options is removed after the packet is drained out of this buffer completely. But, as the buffer space allocation between input and output buffers generally has no significant influence to performance [26], making the input buffer smaller is often impossible.

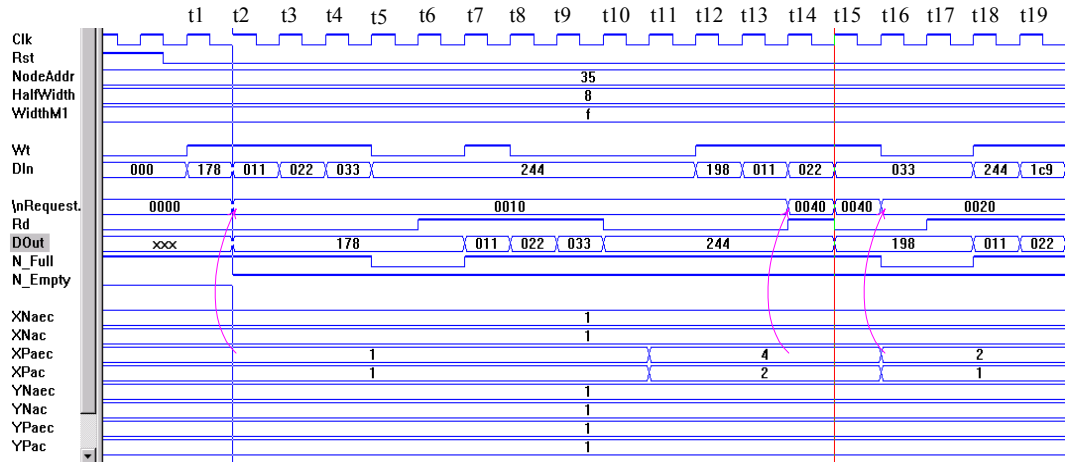


Figure 5.8. Logic verification of the input channel design.

The parallel scheme is verified as shown in Figure 5.8 for the input channel interfaced with the local injector of a node (3,5) in a 16x16 torus. The first packet, whose header is 198 and tail 244, goes to node (7,8) at t1. It can use any of the three virtual channels in XP direction, the Escape, Cyclic or the Adaptive channels. As the escape channel is suggested (XPaec = 1) by the central status collection and optimal path generation circuit sitting on the output sides, it is requested for use. The request is granted (Rd) at t6, possibly as a result of the congestion during t1 to t5. Because the input channel FIFO is always driven, the header can be read out to the output channel through the switch at the same cycle. Another packet headed to (9,8) is decoded at t12, but the request cannot be issued because the previous packet still holds the output port of the FIFO. When the port is released at t14, the request to the Adaptive channel, as suggested by the optimal path generating block, is issued. If the simple design were used, the request would not be issued because the decoding would not be done. If the request cannot be granted due to congestion, a different

optimal path can be generated as a new alternative. As shown at t16, the Cyclic channel (Xpaec=2), rather than the Adaptive channel, is requested and granted at t17.

The input channel interface with the injector is quite complicated as shown in Table 5.3. It can be designed either non-parallel or parallel. The synthesis results are shown in Table 5.4. The parallelization results in a 27% speed up with only 2% chip area over head. Comparing this result with that in Table 5.1, the case of a 4 by 10 input buffer implemented as a circular queue, injector routing takes about 60% of the total chip area of the input channel.

Table 5.4. The cost and speed of two different designs of input channel.

Design	Area (cells)	Timing (ns)
No parallel	10,270	1.30
Parallel	10,485	.94

5.5 Round-Robin Arbiter

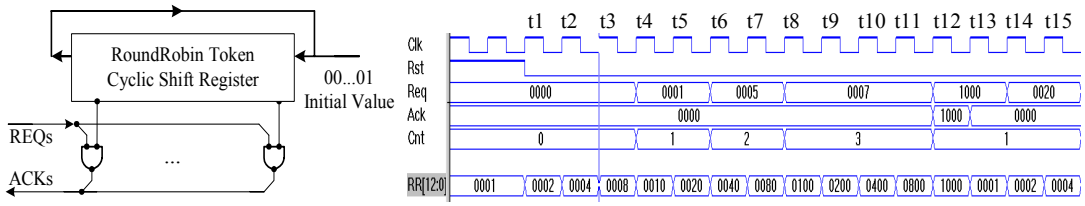
The arbiter is used to resolve conflicting requests for each output virtual channel by packets in input virtual channels (or lanes), and for each physical link by flits in output virtual channels (or lanes). The requests are tabulated by a fixed order in a multiple bit register with each bit corresponding to a specific request. An asserted bit meaning a request is issued from the source (*candidate*). At most one request can be granted at any time – this is defined as a turn being given. If the turn is given in a circular manner, it can be viewed as an exclusive ***token*** cycling around. If the request is in the same position as of the token being asserted, it is called a ***turn-hit***, otherwise it is a ***turn-miss***. The arbiter generates the acknowledgement signal if both of the following conditions are satisfied:

- There is really a request, or the request bit is asserted;
- The turn is exclusively given to this request, or a turn-hit.

Three designs of the arbiter are discussed in this section. Starting from a brief analysis of the requirement for skipping logic in a simple design, we develop a hierarchy to enhance arbitrator performance when skipping is used.

5.5.1 A simple Round-Robin Arbiter

A simple round-robin arbiter can be built with a shift register and several AND gates, as shown in Figure 5.9a. The turn is initialized to give the rightmost request only and then the exclusive turn-tag is cyclically shifted left at each cycle.



(a) (b)

Figure 5.9. A simple round-robin arbiter. (a) Schematic. (b) Logic Simulation and verification.

The arbitration designed by this way is fair, but will be suboptimal in the following situations:

- If there are not many users (candidates) to compete for the resource, then the probability that the token hits the request becomes very low. This leads to either turning the turn-missed request for another resource (if does exist) that may have been already very 'hot', or delaying its service.
- If the arbitrated resource is very non-uniformly 'hot', the asserted requests take a few consecutive bits but nothing in other bits. A miss in the consecutive blank bit may lead to prolonged service for each of the requests.

As shown in Figure 5.9b for 16-candidate arbitration, the request from the lowest-numbered candidate at t4 cannot get serviced for 9 cycles, even when there are no other pending requests at all, and has to turn away at t12.

5.5.2 Arbiter with Naïve Skipping Logic

To eliminate the performance loss of turn misses, the arbiter should keep searching one by one in the cyclic direction, from the last turn hit until another hit. The circuit, as shown in Figure 5.10, consists of a token register (RRT), which keeps the token and the last turn given, and a combinational circuit (SL). This tries to find the nearest asserted request along a certain direction in case there are any asserted bits. If the

nearest request bit is not asserted, it will check the next one until a nearest bit is asserted. The first asserted request will be granted service, and SL will refresh RRT (like a barrel shifter) with the current turn given.

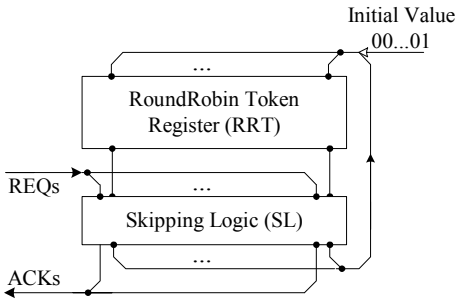


Figure 5.10. A round-robin arbiter with skipping logic.

When the number of candidates being arbitrated increases, the skipping logic will not only take more chip area, but also significantly slow down. The arbiter has two basic operations, matching and loading. The matching is to confirm whether the tabulated request bit, one by one along pre-defined direction starting from the last token position, is valid. If the nearest asserted request is found, the RRT is re-loaded with a new token to specify the next turn-given. If the matching operation on each of the n candidates takes T_m ns and the re-loading takes T_l ns, the arbiter speed is limited by the time T_a,

$$T_a = (n-1)\, T_m + T_l,$$

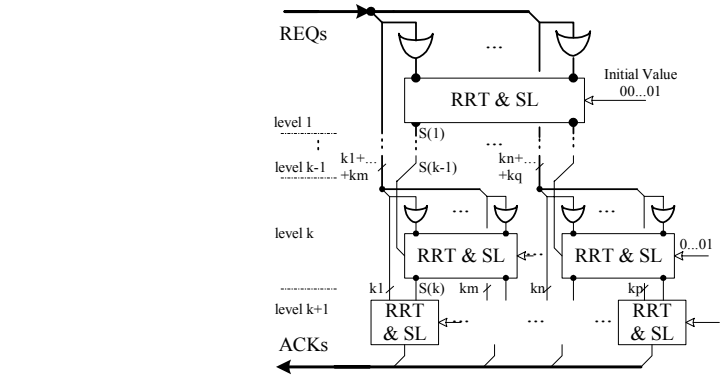
because in the worst case, the arbiter has to skip (n-2) times.

5.5.3 Effective Hierarchic Skipping

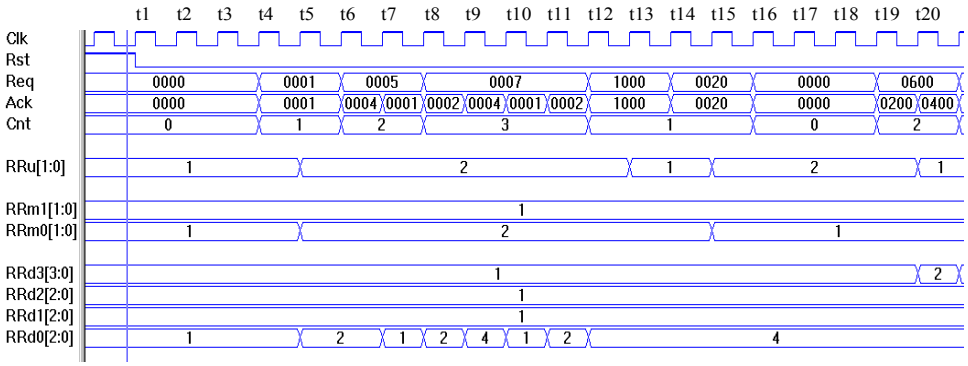
The cycle time of the naïve skipping approach is of O(n), which may be on the critical path of the overall router design. To speed up the arbiter, the arbitrated candidates can be re-tabulated properly in groups and skipped hierarchically, as described in Figure 5.11a.

The hierarchy has a topology similar to a tree. For example, a total number of n = 2m candidates can be partitioned into two groups and each has m candidates to be arbitrated. The top level then just needs to handle 2 requests, and has a 2-bit turn register. Each of these two requests is a reduction of a group, that is, if any of the request bit in the group is valid, the top level corresponding request is set. The naïve skipping logic works independently at each hierarchic level, and at each smaller group of the same level. After

finishing its own operation, one of the lower level groups will get the turn (token) passed from its parent node and its generated turn is passed to its child node. The logic works in a manner of a chain, and the leaf node will finally grant the request.



(a)



(b)

Figure 5.11. An arbiter designed by grouping and hierarchic skipping. (a) Schematic graph. (b) Logic verification of arbitration of 13 candidates.

The logic is verified as shown in Figure 5.11b for 13 candidates partitioned into three levels. Each level has 1, 2, 4 node(s) respectively from top to low level, and the candidates are grouped as {{{4} {3}} {{3} {3}}}. The register RRu, RRm1, RRm0, RRd3, RRd2, RRd1, RRd0 is used to log the token position in each node, and all of them are initiated with a token on the lowest position. At t4, only the lowest numbered request is asserted. After the request is granted, the corresponding token registers RRu, RRm0 and RRd0 are refreshed by shifting the token to the next. All others remain unchanged as the given turns have not been used and there are no requests under their control. RRd0 at t6 is unchanged because the tokenized request is not asserted, but it is skipped to the nearest asserted request, 4, and the token is shifted

to its next position (cyclically), the first position at next cycle. At t7, the request is granted because of the turn hit.

The cycle time of the arbiter (T_{a_g}) depends on the maximum time of its node operations (T_{a_gx}) and the maximum time for a token passing along the chain (T_p).

$$T_{a\_g} = T_{a\_gx} + T_p.$$

T_{a_gx} is much smaller than T_a. Also, T_p can be small if the hierarchy depth is properly designed. Therefore T_{a_g} can also be smaller than T_a.

Grouping and hierarchy skipping can be faster than naive skipping if constrained by the same cost (chip-area). Table 5.5 shows the cost and speed of three different designs for arbiters in the router. To be consistent with each other, all the synthesis optimizations are based on the timing constraint. The partition for the cases with 6, 10, 13 candidates is optimized as {{3} {3}}, {{{3} {2}} {{3} {2}}} and {{{4} {3}} {{3} {3}}} respectively. It can be seen that

- Skipping logic, although necessary, not only increases the chip area cost, but will also slow down the arbiter.
- Grouping and hierarchy skipping is more cost-effective than naive skipping. Although the initiative to speed up is not as much as expected in the case of 13-candidate arbitration, the cost is reduced. To view this in another way, if under the same budget, it can be much faster [9].

Table 5.5. The cost and speed of round-robin arbiters.

# Candidates	Simple		Skipping		Hierarchy Skipping	
	Area (cells)	Timing (ns)	Area (cells)	Timing (ns)	Area (cells)	Timing (ns)
2	80	.20	204	.34		
3	187	.27	444	.49		
6	473	.30	1,908	.72	1,523	.59
10	906	.35	3,875	.88	3,830	.69
13	1,232	.39	7,324	.94	6,524	.76

5.6 Output Channel Design

An output channel consists of a round-robin arbiter, a FIFO and a multiplexer, as shown in Figure 5.12. All data outputs from various input queues targeting the same output channel, see Table 5.3, are multiplexed as the input of the output channel (DIn). A round-robin arbiter resolves the conflicts of

multiple requests (PathReq). The turn (_Rd_Adv) given to a granted input channel simultaneously leads to the reading from the input buffer and writing (Wt) to the output buffer since the input FIFO output is always driven. Following flits of the same packet use the path exclusively. The transmission is controlled by the signals reflecting the status of the two ends, NEmpty for the downstream buffer and NFull for the local buffer. The interface between the output buffer and the shared physical channel multiplexer works similarly. For the special requirement of some cyclic virtual channels, an Empty signal is required to guarantee packet-no-co-residence during the arbitration.

As discussed in section 5.3.2, a central collector is used to collect all the status information from each of the virtual channels and to generate an optimal path for each possible set of requests that can be directly derived from Table 5.3. Optimal path generation is arranged in such a way that duplicated computation will be eliminated as much as possible. As the number of requests for each output channel is the most significant factor to the performance, it is used as the status collected.

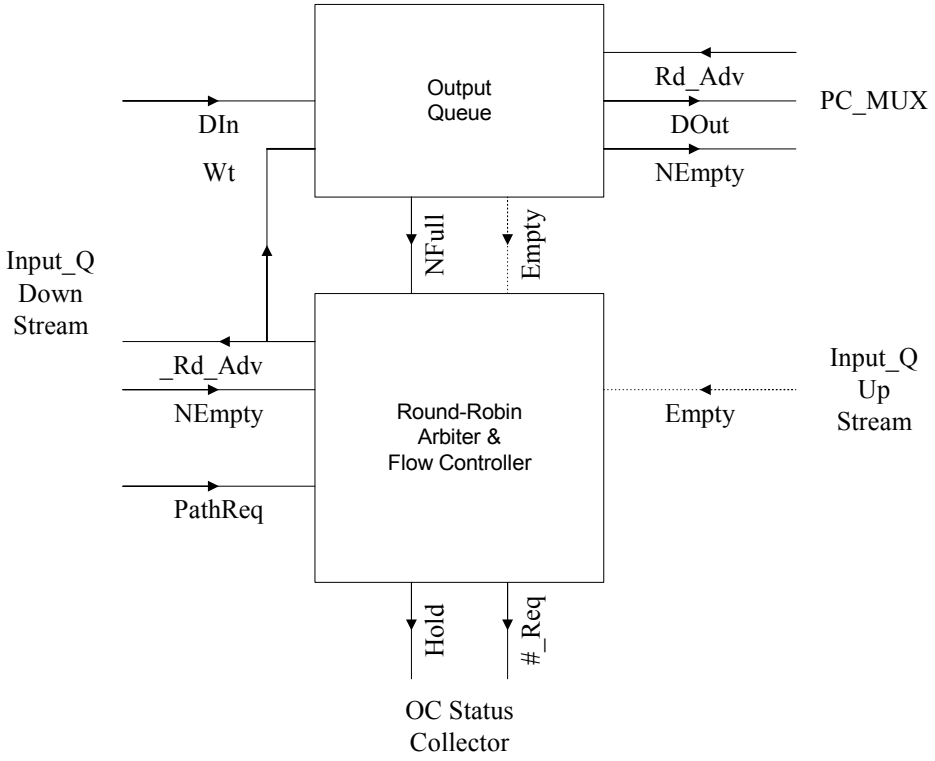


Figure 5.12. Schematic graph of an output channel design.

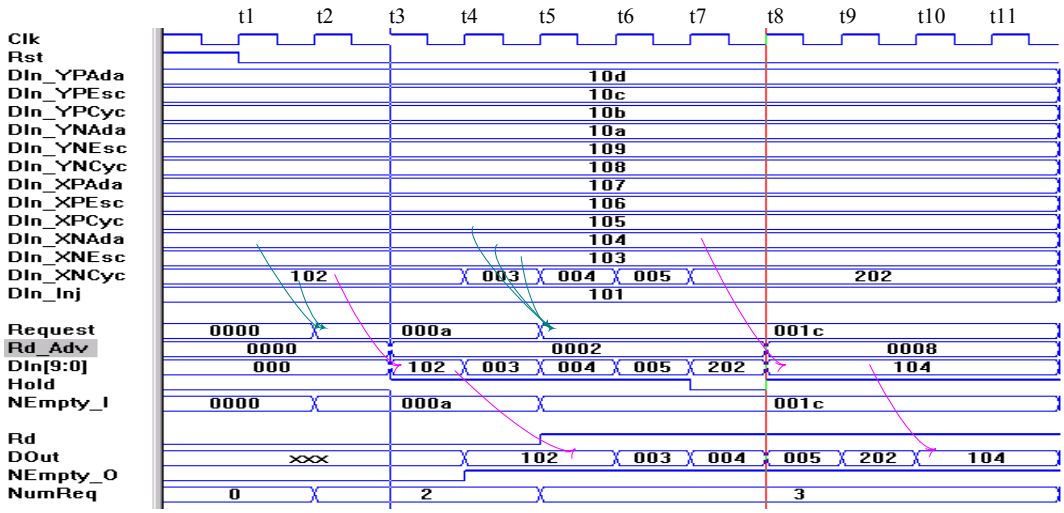


Figure 5.13. Logic verification of the output channel design.

The output channel interface with the local ejector is logically verified as shown in Figure 5.13. At t2, requests from the packets in the adaptive and cyclic input channels along XN are issued, but only the request from the cyclic channel is granted and the header is moved in at t3. The request for using a shared XN link is granted at t5. There are three packets in the escape and the adaptive channels along XN and the cyclic channel along XP that issue requests. None of them is granted until t8 because the connection to this output channel is already being held. At t8, why does the far ordered request from the adaptive channel get served rather than that from the escape channel, the nearest to the just served request? This is due to the hierarchic skipping used. As discussed in last section, the adaptive channel along XN is in a different group from other channels along XN. The top-level arbitration will shift the token to another group containing the adaptive channel, if the group has any request asserted and its next lower group is already served.

Table 5.6 shows the synthesis results for the output channel interface with the ejector and the central status collector. Compared with Table 5.1 and Table 5.5, we can see that the big multiplexer takes up to 30% of the total chip area of the ejector channel. We can also see that the central optimal path generator, although it has eliminated duplicated logic and parallelized the comparison of status, is still the slowest among all the parts of the router. But, as we will discuss shortly, it is not on the critical path of the overall router design.

Table 5.6. The cost and speed of an output channel design.

Design	Area (cells)	Timing (ns)
Ejector Output Channel	15,083	.99
Central Optimal Path Generator	24,616	1.29

5.7 Router Design Verification and Evaluation

The router is verified and synthesized and its cost and speed are extracted.

5.7.1 Overall Logic Simulation and Verification

A test case for a 16x16 torus is illustrated in Figure 5.14. We assume that before cycle t1 there is no packet in the network and the router is in its initial state. There are a total of four packets involved in the simulation of the router, whose local address is (3,5). Packets P1 and P3 are injected from the local node with destination node addresses (7,8) and (6,6) respectively. Packets P2 and P4 are passed from node (2,5) and are heading to node (5,a) and (9,2) respectively.

At time t1, P1 is ready in the injector and is transferred (dWt_Inj=1) via the injector interface to the input buffer. The header is extracted at next cycle and an optimal path is requested at t2. Because the higher priority channel is available, the request is granted and the header flit (178) is transferred to this channel at t3. It takes exactly two cycles for a header to be transferred from the input buffer to the output buffer. While the following data flits are being transmitted through the path, the header flit in the output buffer then requests the use of physical link (XP) at t4. The request is granted (uWt_XP=1) at t5 and the packet header finally leaves the router.

At t6, the P2 header appears at the input physical link (XP) by winning the multiplexer arbiter within cyclic virtual channel (dWt_XP=1) in downstream node. It can use any of the three virtual channels to proceed, but as the cyclic channel is already in use by P1, the escape channel is requested at t7. At the same time, as the P2 header is moved in, the buffer now is not empty. Another cyclic fully adaptive channel buffer is still empty with signal dE_XP = 10. The P2 request is granted and its header is moved in at t8. It competes for physical link XP at t9 and finally gets out (dWt_XP = 10) at t10.

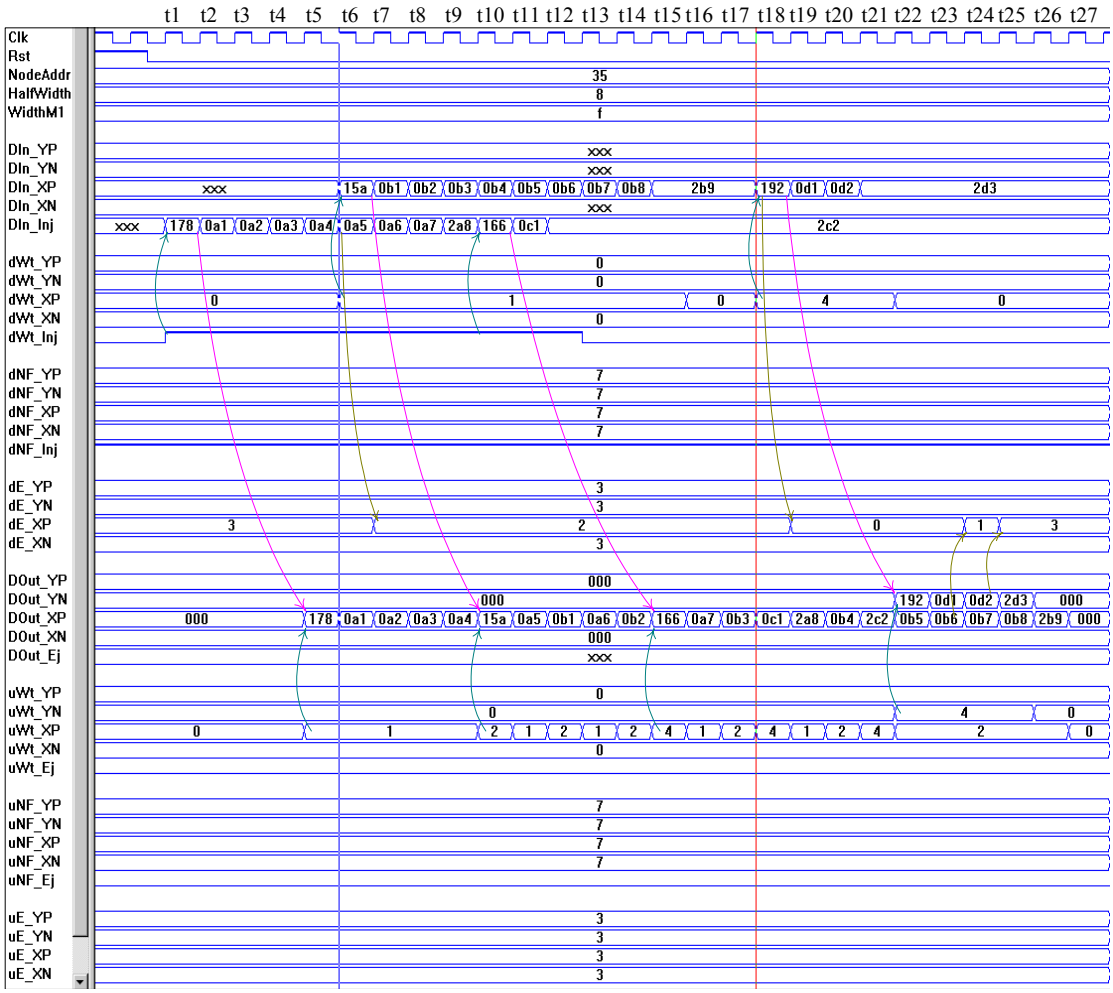


Figure 5.14. Logic simulation of the F_DynBal router.

At t10, the P3 header appears at the injector interface. Because two of the three virtual channels are being used, it can only use the adaptive channel to proceed. Its header competes for use of the physical link (XP) at t14 and gets out (dWt_XP=100) of the router at t15.

The three packets hold the three virtual channels from t15 to t18, no one finishes at t18. Hence, packet P4 from the downstream adaptive channel (dWt_XP = 100) at t18 cannot proceed via the XP link. But it can proceed in YN direction by using the adaptive channel instead, as promised by the algorithm. The request is granted at t20 and the header is transmitted into the output buffer. The header is granted the physical link YN at t22 (dWt_YN = 100). The four packets get out of the local router at t19, t21, t25, and t26 respectively.

5.7.2 Synthesis and Critical Path Analysis

The critical path of the overall router design is shown in Figure 5.15. It starts from the input FIFO read, continues to the FIFO data output, and then through the switch to the input port of the output FIFO.

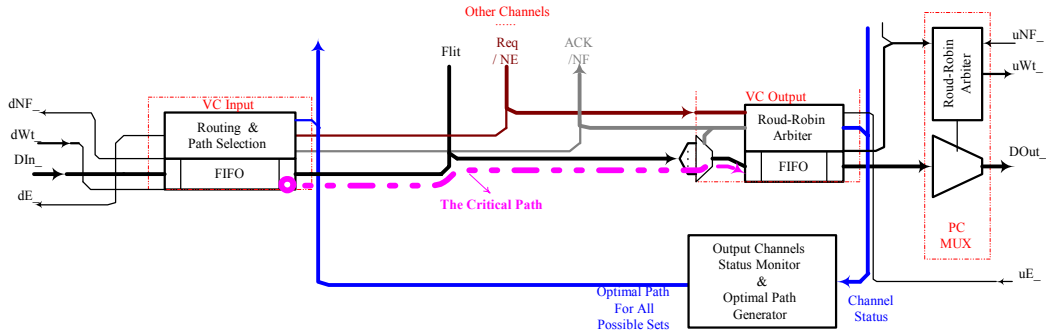


Figure 5.15. The critical path of the adaptive router design.

The complicated control logic of adaptive routing is no longer on the critical path. This has benefited from the design of components and the state-of-the-art ASIC library used. The parallelization of the routing path generation and optimal path selection removes them from the critical path. Central status collection and comparison, although the slowest component, does not become the bottleneck. The data transmission on the critical path begins when the data path is granted. Flits in the input buffer slots are shifted towards the buffer head, while the flit at the head is driven through the switch, a multiplexer controlled by the output channel arbiter. Finally, the flits fill into a proper slot of the output buffer.

The time used by the multiplexer is about 37% of the critical path, while the FIFO operation takes about 63%. Smaller buffer size may reduce the critical path timing (if implemented in a circular queue), but less than 2 slots per FIFO will introduce bubbles and will cause performance loss. Making the switch smaller by cascading may also benefit the timing, but it is not an option for fully adaptive routing.

5.7.3 Cost and Speed of Routers

The routers implementing TRC, DynBal, Duato's PA (or ParAdapt), and Duato's FA (or F_ParAdapt) routing algorithms can be designed by modifying the design of F_DynBal. The chip area and clock timing for each of these wormhole routers are shown in Table 5.7.

Table 5.7. The cost and speed of wormhole routers.

Algorithm	Number of Lanes	Buffer Size (flits)	Area (cells)		Timing (ns)
			Cell	Total	
TRC	9	108	91,371	139,113	1.18
DynBal	9	108	103,634	155,532	1.25
Duato's PA	9	108	101,750	154,467	1.25
F_DynBal	13	104	147,160	229,662	1.47
Duato's FA	13	104	143,667	223,158	1.47

The cost and speed of other routers can be derived either from previous work or by comparisons. The static balanced scheme (T3D-like or StaBal) is supposed to have the same cost as TRC if the routing lookup table is constructed by SRAM (either on-chip or off-chip), which shifts the cost to local node from router as shown in a previous work [56]. The key problem is that, if we assume the same timing for both TRC and T3D-like, we should construct the LUT as registers – even an on-chip cache would force the routing time much longer than 1.18 ns. The same work also developed a scheme to tackle the timing bottleneck and on-chip space shortage related to very large buffers required by VCT. If the basic module of a VCT router in this work is re-synthesized using the LSI G11-p library, it costs 87,539 cells area (a total of 45 flits of register buffer is required, not counting the SRAM) and clocks at 1.29 ns. The hybrid scheme (FD_DynBal) should have the same clock cycle as F_DynBal because of the almost same routing strategy – the only difference is that the large drain-off buffer handling will not be on the critical path if the buffer is the same size as in F_DynBal. It will, however, take a little bit more chip area because of the extra comparison logic of available buffer space, just as the hard-wired dateline causes DynBal to use more space than Duato's PA. The other hybrid router, T3E-like or FD_StaBal will have the same clock cycle and take less control space than F_DynBal because of the simple control on non-fully-adaptive VN. It will take more buffer space if the LUT is implemented as registers.

5.7.4 Design Strategies and the Evolution of Micron Technology

In Table 5.8a, routers implementing three algorithms are compared by their speed under two different processes. Table 5.8b lists the relative slowdowns of adaptive routers than TRC. The effect of the process

used is significant, and contributes to more than four times speedup. But why is there more speedup in the new model, which implies a preference for adaptivity?

Table 5.8. Effect of design strategies and process technology.

Algorithm	Timing (ns)		Speedup
	Cost-speed Model on .8 micron Gate Array	New Model on .25 micron LSI ASC	
TRC	5.30	1.18	4.49
DynBal	5.90	1.25	4.72
F_DynBal	7.10	1.47	4.83

(a)

Algorithm	Slowdown than TRC	
	Cost-speed Model on .8 micron Gate Array	New Model on .25 micron LSI ASC
DynBal	11%	6%
F_DynBal	34%	25%

(b)

The different preferences for adaptivity are caused by disparate implementation details between these two models rather than the technology used.

1) More complicated and fair arbitration, Round-Robin is used in this work, rather than the simplest Highest-First in the cost-speed model, which resulted in starvation;
2) Buffered wormhole is assumed in this work, which mean the buffer depth will affect the speed if the buffer is implemented as a circular queue, while this is not modeled in the cost-speed model;
3) Hierarchic skipping, with some parallelization within each hierarchy, is used to speed up the complicated arbiter, but this is not a concern in the cost-speed model;
4) Decoding the header is parallelized with the input buffer operation to isolate the routing control from data transmission, which is not taken into account in the cost-speed model because of a shallow buffer.
5) Decoding the header is parallelized with selecting an optimal path, while the routing and path selection is done in order in the cost-speed model.

Items 1 and 2 will significantly slow down a design, but the adaptive router is more affected because of the additional options needed to arbitrate. The followed items are the enhancements for speed or to deal with the problems introduced by the first two items. It has been clearly shown that without sophisticated

implementations, the adaptivity would lead to a much slower router than that has been achieved by our design.

The contemporary evolution of VLSI process technology offers a challenge to the router design [73]. The additional gates in a single chip give a chance to implement buffered wormhole routing and complicated control. On the other hand, as faster PEs becomes available, the router design is forced to be more sophisticated.

5.8 The Cost-effective Router

In this section, we compare and contrast the performance to the area cost and timing of each routing scheme. We draw basic pictures of cost-effectiveness and present a plausible SoC multi-computer based on the technology development.

5.8.1 Performance and Cost of Routing

Combining the cost-speed analysis of the routers with the performance evaluation results from section 4.4, it emerges that the dynamically balance scheme, especially the dynamically balanced fully adaptive buffered-wormhole routing (F_DynBal) algorithm is the most cost-effective and flexible.

- Compared to TRC and static optimization of virtual channel assignments, dynamic balancing performs better with little cost. Although DynBal is 6% slower than TRC, it can have up to 40% more throughput per cycle, resulting in 32% more throughput per ns, with only 12% more cost in chip area. It can have up to 24% more throughput per ns than T3D-like under local traffic. F_DynBal has up to 15% and 31% more throughput per ns than T3E-like and T3D-like, respectively, in case of large packets.
- Compared to adaptive selection of virtual channels based on the cycle-tolerance theory, dynamic balancing performs better with less than 2% added cost. DynBal can outperform Duato's PA by up to 27% and F_DynBal can be up to 8% better than Duato's FA, in the case of big ratio of buffer size to packet size.
- Fully adaptive routing performs better under more comprehensive workloads and is more cost-effective. For example, under perfect shuffle, F_DynBal outperforms TRC, DynBal by 126%,

57% more throughput per ns with only 65%, 47% more cost respectively. ??????????????????????

5.8.2 Cost-Effectiveness of Routers under Uniform Traffic

To be independent of the technology used, we assume no on-chip SRAM is used in the router control path. This is especially important for the statically balanced schemes, because otherwise the LUT access time will prolong the routing too much for the assumptions made in section 5.7.3 to be valid. Figure 5.16a shows the cost effectiveness of the four non-fully adaptive routers. The on-chip register implementation of the LUT in the T3D-like router takes too much of the router space (an address-dependent design will reduce the size of LUT, but each router must be specifically configured). Implementing the LUT in on-chip SRAM may save gates, but will significantly slow down the router. As discussed earlier, the routing in T3D-like (also for subset of T3E-like) can be done locally at the packet source node. Figure 5.16b shows the case after the LUT cost is shifted to local node. But we should keep in mind:

1) Source routing will force the network size $1/2^n$ smaller than otherwise, if the width of packet header is fixed – because the virtual channel used for each dimension (n) must be specified in the header tags. Hence, it limits the scale of the network. ????this only takes one bit per dimension???????????

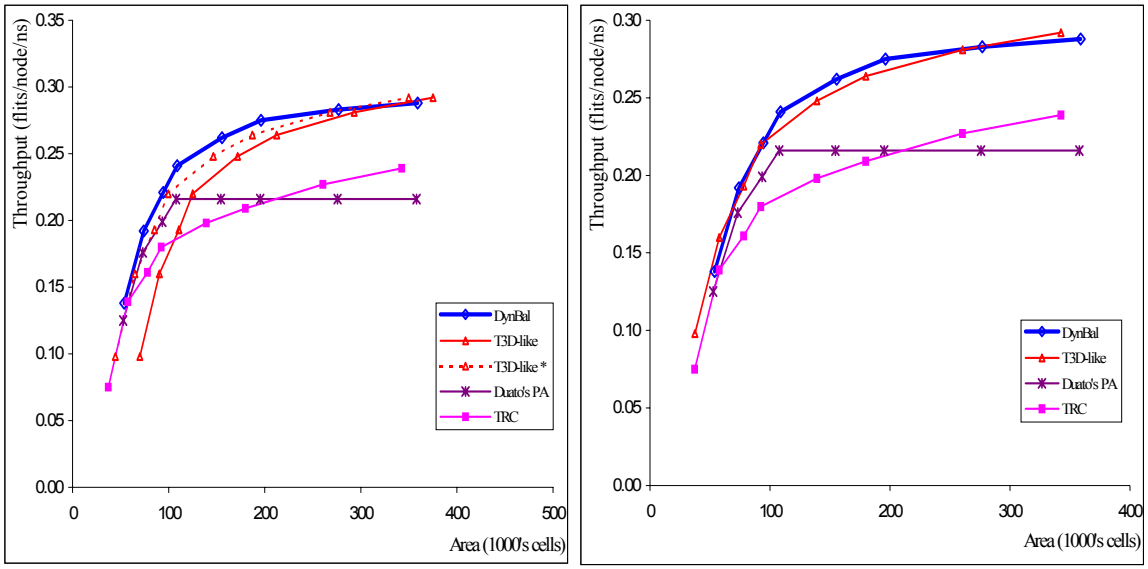
2) If the LUT is implemented off-router in hardware, either it takes local memory or NIC gates, it is shifting the area cost from router to local node. The system cost thus remains the same.

3) If the LUT is implemented totally by software, assuming the LUT can be preloaded during initialization, it will cost more cycles to get the routing done.

4) In the off-router scheme, either hardware or software implementation will complicate the packetizing process. The significant effect on the network performance is to increase the static workload latency. For example, it may add several ns to get the VC tags from the LUT for each dimension and then append them to the packet header. For the static workload used in this research, the packetizing latency is significant, especially for small packets where the network transfer latency is relatively low.

???????????? Compared with the performance evaluation (Figure 4.9), although T3D-like does not perform as well as DynBal, after shifting the cost of LUT, it has the similar effectiveness as DynBal in this case. The TRC, although having a fast clock, is still worse than DynBal because of its unbalanced use of

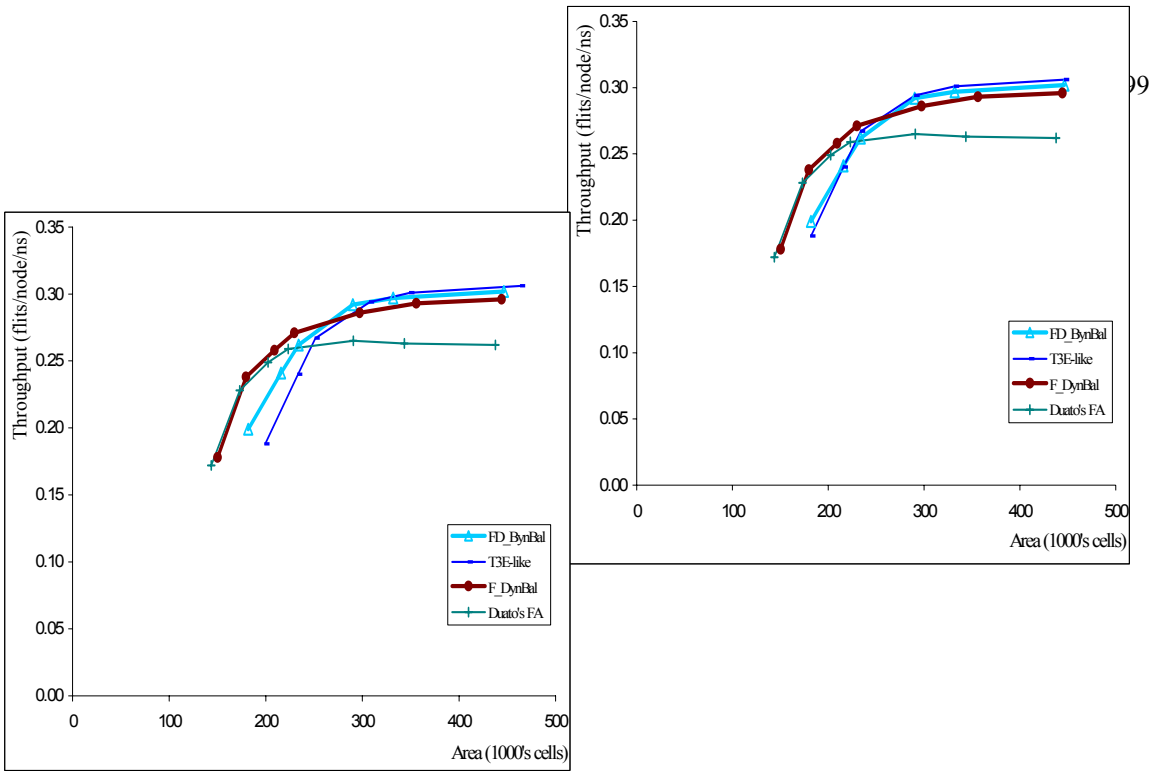
virtual channels. Duato's partial adaptive routing is worse than DynBal because of both the bottleneck and bubbles.



(a) (b)

Figure 5.16. Cost-effectiveness of non-fully adaptive routing algorithms under uniform dynamic workload. (a). LUT embedded inside router as on-chip registers (*: address-dependent LUT design). (b) LUT off-router. (Under environment: total 9 lanes. 8-flits-packet; each cell takes 11.34 square micrometer).

Figure 5.17 shows the cost-effectiveness of the four fully adaptive algorithms, with and without LUT implemented as registers. The T3E-like has similar cost-effectiveness to F_DynBal, but Duato's fully adaptive algorithm is worse than F_Dynbal, especially when there is enough space to construct a larger buffer. The two hybrid VCT and WH routers have a little bit better performance than the pure WH routers, but only if there is enough space and the performance difference is quite small. If the area budget is less than 280K cells, then the pure WH routers even outperform the hybrid ones. Because the fully adaptive channel (drain-off buffer) must guarantee enough space for the biggest packet in a system, the chip area required by the hybrid schemes will increase for bigger packets. This not only makes the hybrid scheme less flexible, but also makes it less attractive to use, as discussed later.



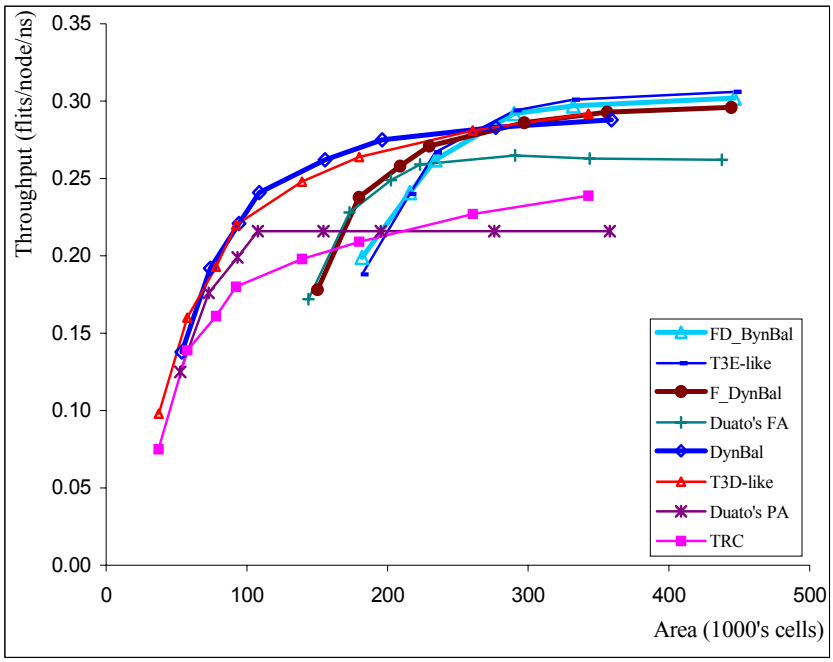
(a) (b)

Figure 5.17. Cost-effectiveness of fully adaptive routing algorithms under uniform dynamic workload. (a). LUT embedded inside router as on-chip registers. (b) LUT off-router. (Under environment: total 13 lanes. 8-flits-packet; each cell takes 11.34 square micrometer).

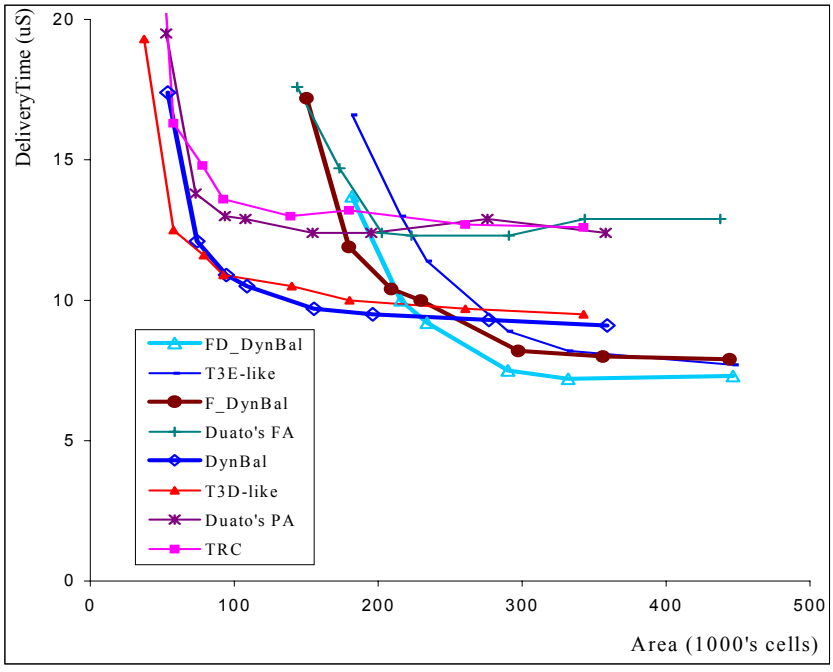
By combining Figure 5.16 and Figure 5.17 together as Figure 5.18a, we can compare the cost-effectiveness of the non-fully adaptive routers with that of fully adaptive routers. Although the LUT implementation in T3D-like and T3E-like depends on many things, we do not add overhead in the following discussion unless explicitly specified. The fully adaptive routers can out-perform non-fully adaptive routers only if the area budget is greater than 280K cells.

It would be interesting to compare Figure 5.18a with Figure 4.9. Although the performance simulation shows that fully adaptive routing has much higher throughput than non-fully adaptive, it becomes much less attractive when the cost and speed are considered. Maybe this is one of the reasons that the adaptive routing has not been widely used in commercial machine [25].

Compare with the dynamic workload performance, the fully adaptive routers get more performance under static workload, as shown in Figure 5.18b. F_DynBal has as low as 60% of the time for Duato's fully adaptive router to drain off the packets, and FD_DynBal uses nearly 20% less time than the T3E-like router – not counting the extra latency introduced by off-router routing processing.



(a). Dynamic workload.



(b). Static workload.

Figure 5.18. Cost-effectiveness of routers under uniform workload with 8-flits maximal packets. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. Each cell takes 11.34 square micrometers).

If the extra time (taken by a local node to access LUT and appending to packet header) were assumed to be 10 ns, the time for T3D-like and T3E-like would be much worse. A 256-packet data matrix vector in each node would take extra 2.56 us, which makes T3D-like take 25% more time than DynBal and T3E-like take 40% more time than FD_DynBal.

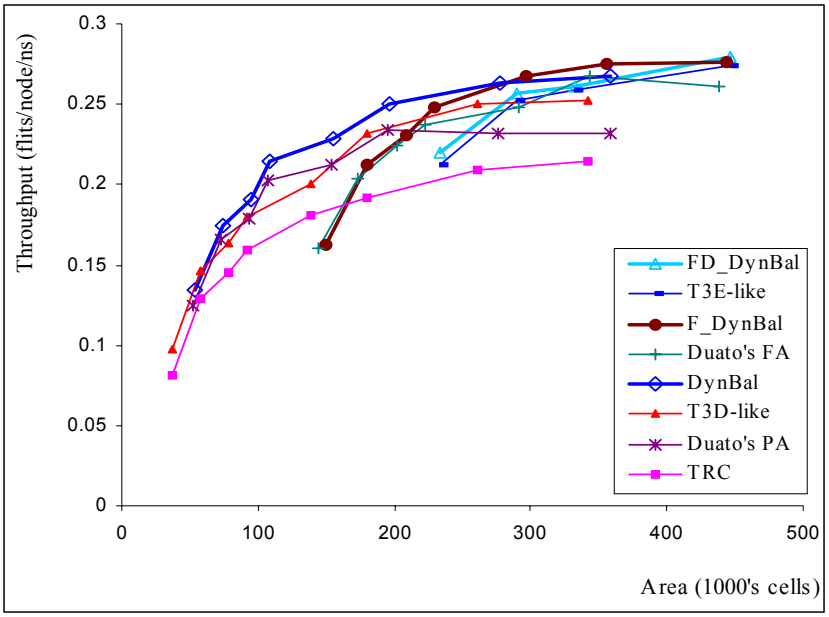
Figure 5.19 illustrates the cost-effectiveness of routing algorithms when packets get bigger. For dynamic workload, the hybrid schemes (both FD_DynBal and T3E-like) not only require more space than in smaller packet systems, but also perform worse than F_DynBal. The difference between F_DynBal and Duato's FA, DynBal and Duato's PA becomes smaller than that for smaller packets because of fewer bubbles for the latter. The DynBal more visibly outperforms T3D-like because the statically balanced nature of the latter does not have enough flexibility to select routes when a blocked packet holds multiple links. For Static workload, the F_DynBal performs more poorly than DynBal because the switching between dimensions causes congestion.

5.8.3 Cost-Effectiveness of Routers under Local Traffic

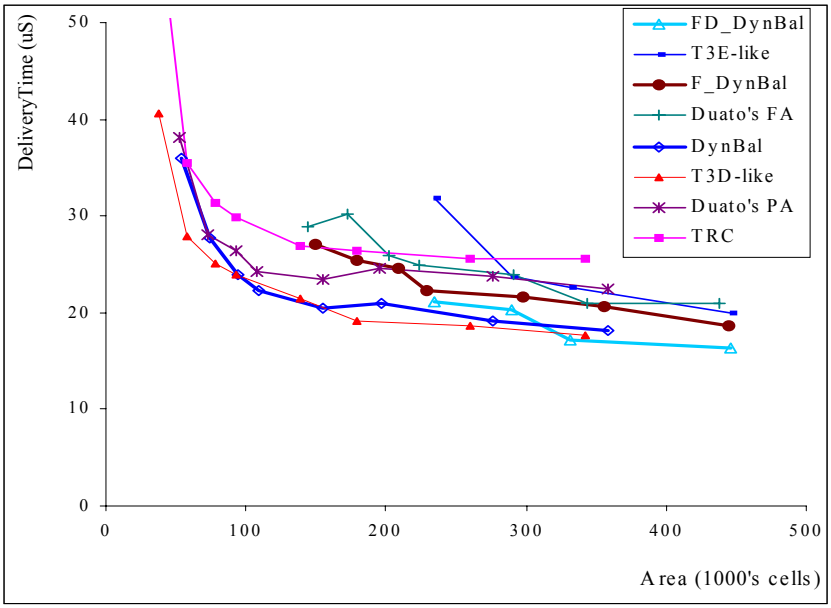
Local traffic can be distributed locally either uniformly or non-uniformly. Random near communication will distribute the packet randomly within a distance less than the average network path length, while the barrel-shifter-like shifting will distribute packets non-randomly within this range. T3D-like routing and sub-routing of T3E-like use globally optimized LUTs so that they do not work well for local traffic.

Figure 5.20 illustrates the cost-effectiveness of routers under random near traffic for both dynamic and static workloads. The performance of DynBal is the best for both dynamic and static workloads. Also, the difference between DynBal and T3D-like is more significant than that under uniform traffic.

Figure 5.21 shows the evaluation of routers under barrel-shifter-like (diagonal shifting) traffic. Because of the non-random nature of the traffic, fully adaptive routers perform better than non-fully adaptive ones. For dynamic traffic, F_DynBal is the best and FD_DynBal outperforms T3E-like by more than 10%.

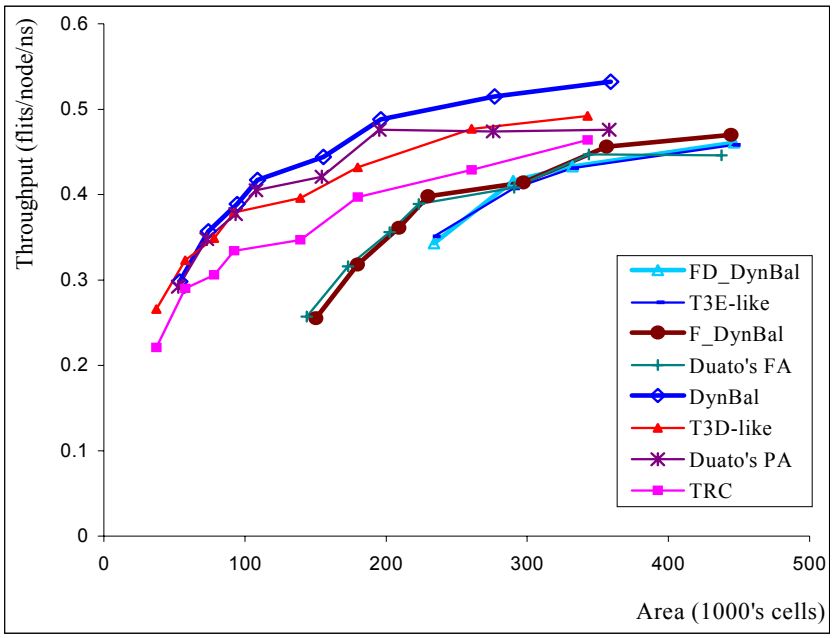


(a). Dynamic workload.

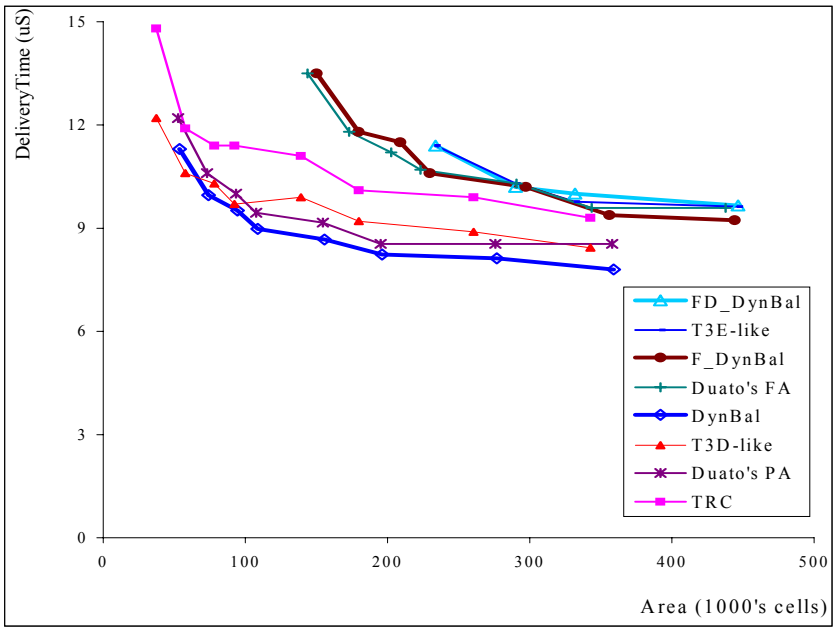


(b). Static workload.

Figure 5.19. Cost-effectiveness of routers under uniform workload with 16-flits maximal packets. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. Each cell takes 11.34 square micrometer).

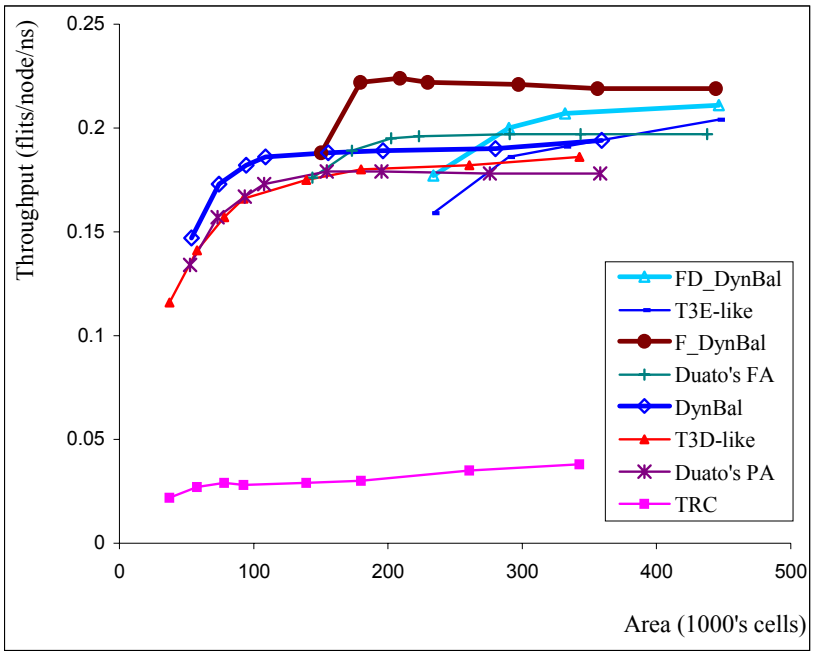


(a). Dynamic workload.

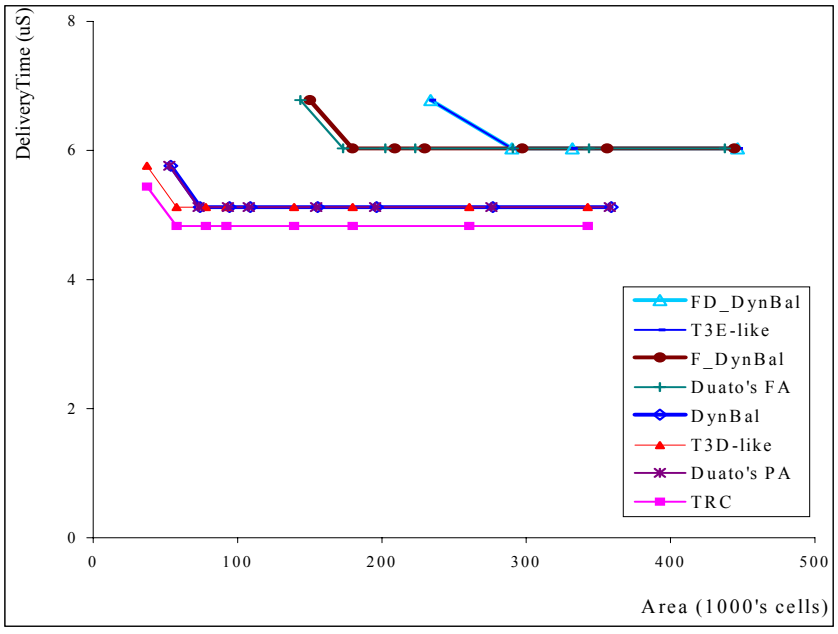


(b). Static workload.

Figure 5.20. Cost-effectiveness of routers under random near traffic. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. 16-flits-packet; each cell takes 11.34 square micrometer)



(a). Dynamic workload.



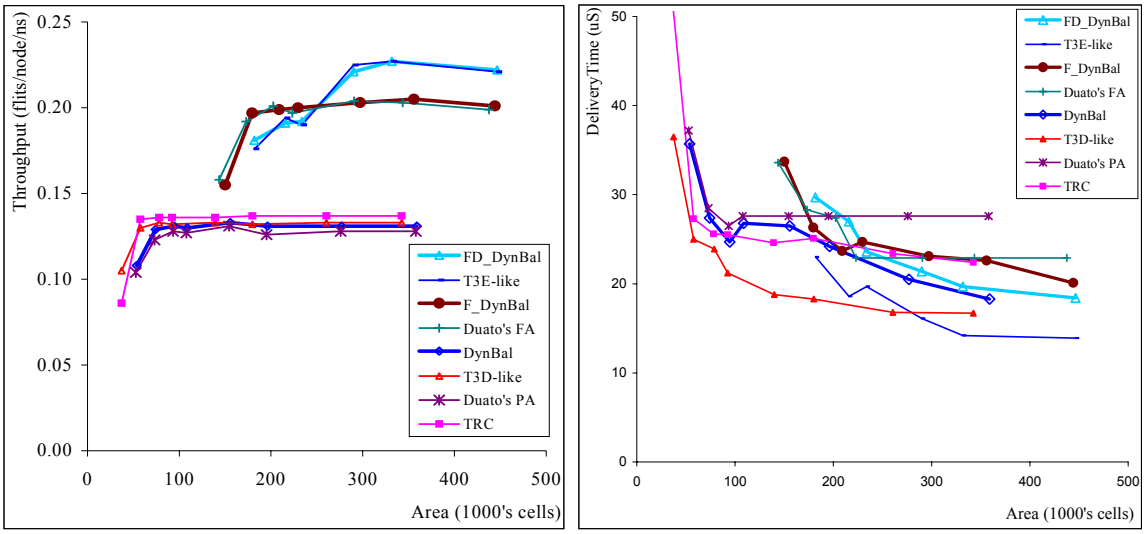
(b). Static workload.

Figure 5.21. Cost-effectiveness of routers under local workload -- shifting. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. 16-flits-packet; each cell takes 11.34 square micrometer).

The static workload delivery time for each algorithm in Figure 5.21 is constant except the case of the one-flit input buffer (which will stall the physical link during two-cycle header routing).

5.8.4 Cost-Effectiveness of Routers under Non-uniform Traffic

The non-uniform traffic can be classified as non-random concentrated and random concentrated. Bit reversal and dimension reversal are typical patterns of the former, and have been used in previous works[26]. Random bit vector will be used as an example of latter in the following discussion.



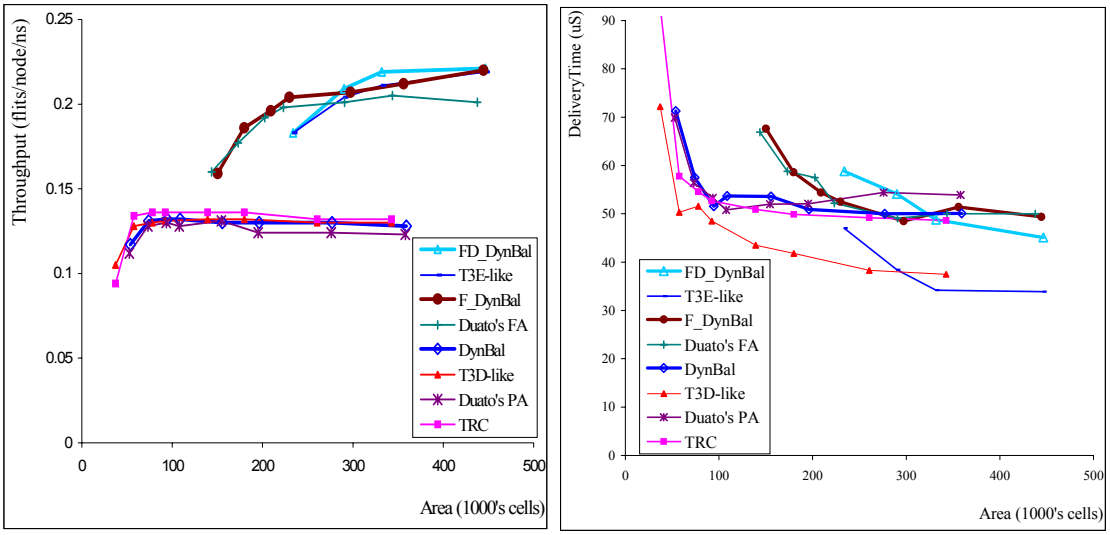
(a). Dynamic workload. (b). Static workload.

Figure 5.22. Cost-effectiveness of routers under bit-reversal traffic with 8-flits maximal packets. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others; each cell takes 11.34 square micrometer).

The evaluation of routers under bit-reversal traffic with (maximal) 8-flits packet is shown in Figure 5.22. For the dynamic workload, the difference among algorithms in each set--non-fully adaptive, WH fully adaptive, and hybrid--is nearly invisible. Hybrid routing gets the highest rate of accepted traffic, followed by WH fully adaptive routing, and then the non-fully adaptive routing (if the budget available to implement each algorithm). The fully adaptive router performs better because it offers an optional path in another

dimension when congestion happens in one dimension. The hybrid scheme may drain out the packet behind a blocked packet and yet may not cause follow-up congestion in another dimension by making excessive turns. For the static workload, both T3D-like and T3E-like routers look like they have significant lower performance. But if the source packetizing process latency is considered, there is not much improvement in the delivery time.

When the packet size gets bigger, the difference between hybrid and WH fully adaptive routing is reduced, and the FD_DynBal performs better than T3E-like, as shown in Figure 5.23.

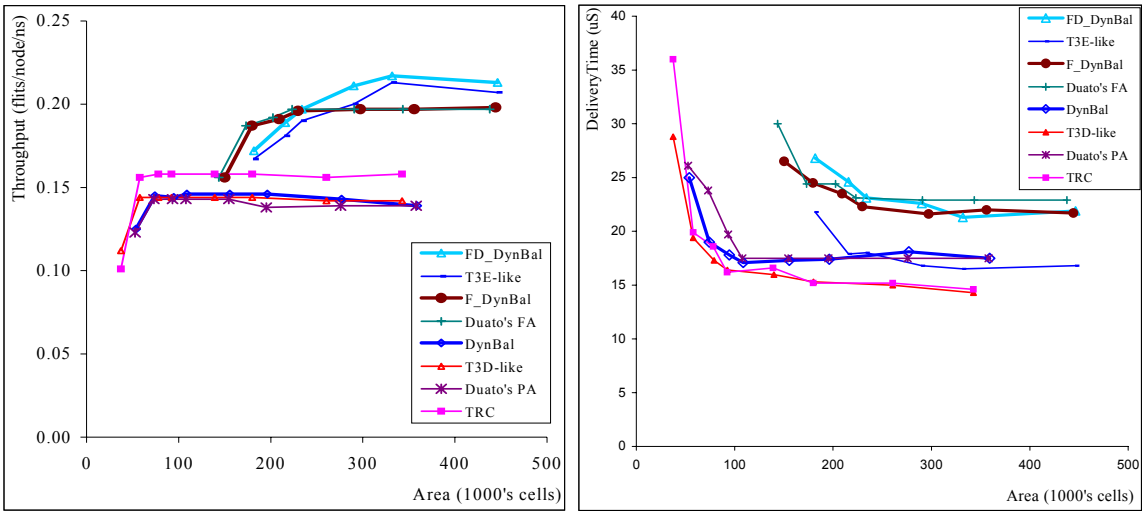


(a). Dynamic workload. (b). Static workload.

Figure 5.23. Cost-effectiveness of routers under bit-reversal traffic with 16-flits maximal packets (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. Each cell takes 11.34 square micrometer).

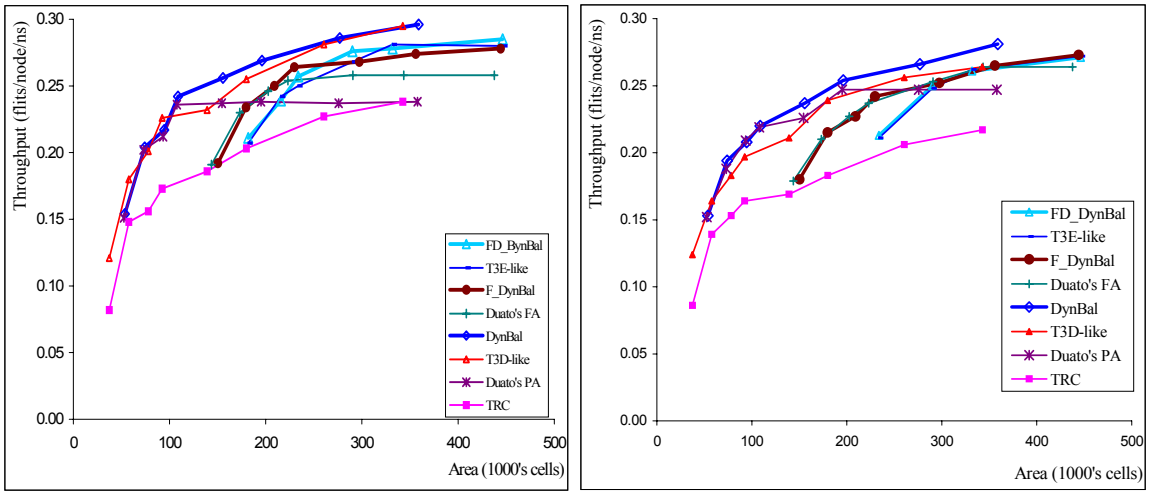
The cost-effectiveness of routers under dimension reversal traffic is shown in Figure 5.24. Compared with Figure 5.23, the T3E-like router as well as the other adaptive schemes, has to use more time than TRC for static workload delivery.

Figure 5.25 shows the evaluation of routers under random bit vector dynamic traffic for packet sizes of 8-flits and 16-flits. Compared with Figure 5.18b and Figure 5.19b, the effect of this kind of traffic to the routers is similar to random (uniform) traffic – DynBal performs the best and the hybrid schemes cannot compete with pure WH fully adaptive routers when packets get bigger.



(a). Dynamic workload. (b). Static workload.

Figure 5.24. Cost-effectiveness of routers under dimensional reversal traffic. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. 8-flits packets. Each cell takes 11.34 square micrometer).



(a). 8-flits packet. (b). 16-flits packet.

Figure 5.25. Cost-effectiveness of routers under random bit vector dynamic traffic. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. Each cell takes 11.34 square micrometer).

5.8.5 Cost-Effectiveness of Routers under Synthetic Traffic

It has been shown that the DynBal algorithm works better for random traffic (uniform, local and some random concentrated non-uniform) and the fully adaptive algorithms work better than DynBal only for non-random traffic. For local traffic and big packet workloads, DynBal outperforms T3D-like. For big packet non-random workloads, the pure WH fully adaptive routing algorithm can achieve the same performance as hybrid scheme. During a specific period of time, the system may concentrate on non-random communication, but in a long run, the traffic can be viewed as uniform. The synthetic workload used in the following is between these two extremes – it is dominated by non-random traffic, but also has some random and local communications. The scenario can be stated as:

1) Broadcasting fixed data to each node (one-to-all),

2) Parallel computation on the local data which may need some local communication to near neighbors as well as some predominantly non-random communication, like dimension reversal and bit reversal, and

3) May need some communications to globally random nodes, and

4) Send local data to a particular node (all-to-one).

Let's assume that the communication patterns are selected randomly with specific ratios: one-all communication takes 5%, global random 5%, random near 10%, and both bit reversal and dimension reversal each takes 40%.

The evaluation of routers for maximal 8-flit packets is shown in Figure 5.26. The relative performance is a hybrid of previous evaluations. The hybrid routers outperform pure WH fully adaptive routers, and the latter outperform non-fully adaptive ones. The difference is smaller than in purely non-random traffic. Compared with the previous results, the point at which the fully adaptive router outperforms non-fully adaptive ones is shifted left because the latter does not work well for non-random traffic. The DynBal router outperforms the T3D-like, and the FD_DynBal router outperforms T3E-like due to the poor performance of T3D-like and T3E-like under local traffic. The DynBal and F_DynBal outperform Duato's PA and FA, respectively, when more space available due to the bubbles in the latter set.

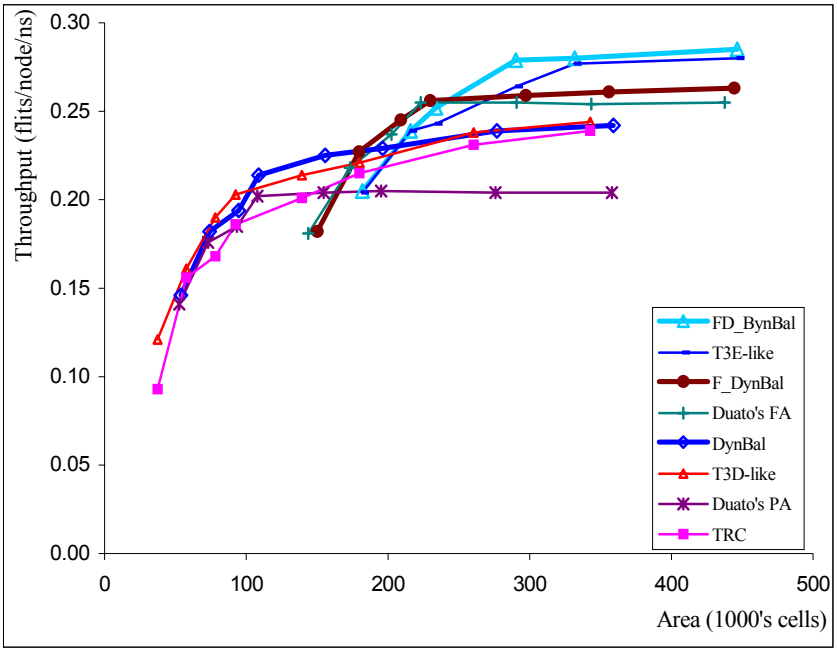
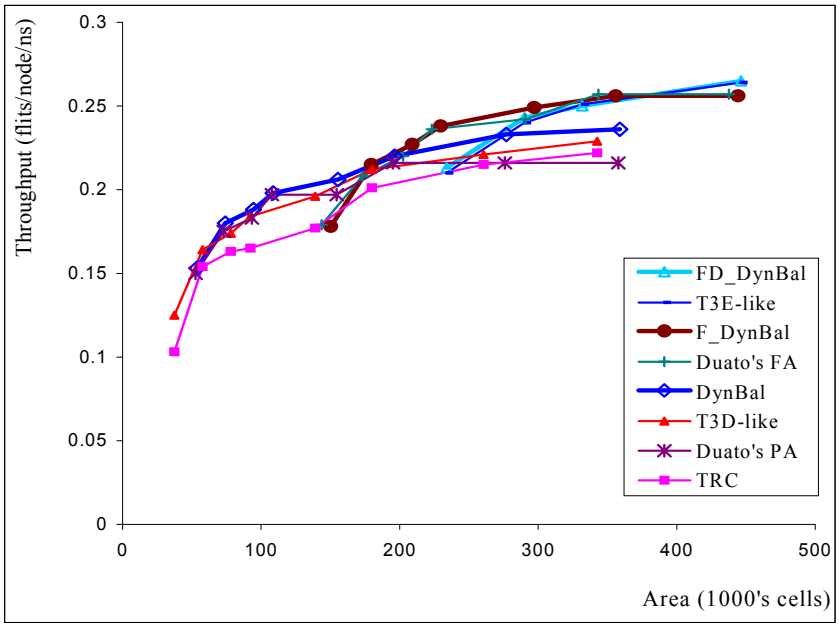


Figure 5.26. Cost-effectiveness of routers under Synthetic dynamic traffic with 8-flits maximal packet. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. Each cell takes 11.34 square micrometers).

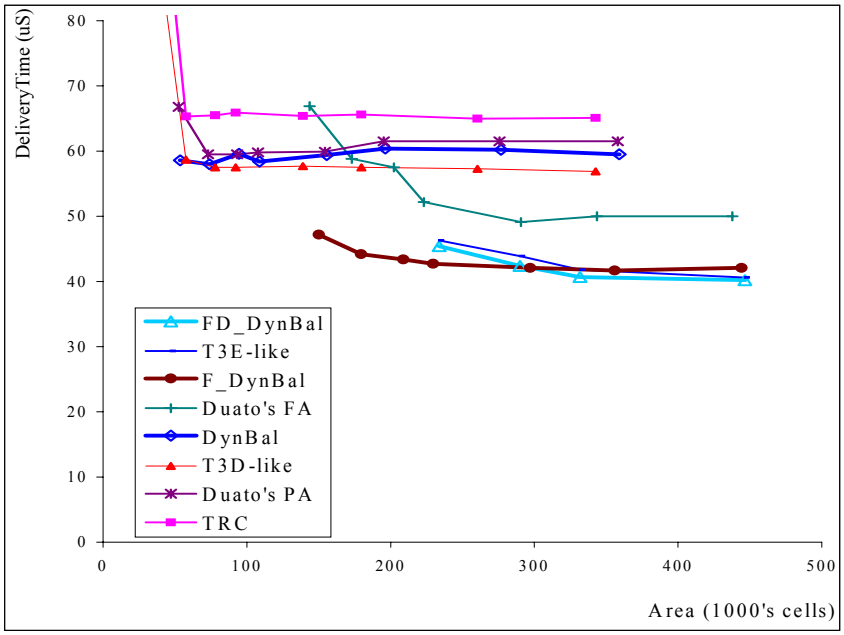
When the packet gets bigger, F_DynBal outperforms all others if over 180K cells are available, as shown in Figure 5.27. The difference between the hybrid router and the WH fully adaptive router is reduced. Compared with the previous evaluations for both non-random and random traffic, the FD_DynBal also has almost the same static workload delivery time as T3E-like – even not considering that the T3E-like would introduce extra latency when accessing LUT and packetizing.

5.8.6 The Cost-Effective Router Design

We have presented the cost-effectiveness evaluation of routers by assuming a single lane per virtual channel. As mentioned in previous work [36], doubling the number of lanes will slow down the router speed about 20% besides almost double the chip area taken – buffers take much of the space especially in buffered wormhole routing. Considering the performance evaluation shown in Figure 4.7, it is clear that more lanes can only drive a design to being less cost-effective



(a). Dynamic workload.



(b). Static workload.

Figure 5.27. Cost-effectiveness of routers under Synthetic traffic with maximal 16-flits paket. (Under environment: total 9 lanes for TRC, Duato's PA, T3D-like and DynBal, 13 lanes for all others. Each cell takes 11.34 square micrometers).

The original intention for developing the three routing algorithms was to find a low cost yet high performance router design. DynBal was developed to contend with TRC, Duato's partial adaptive routing and T3D-like routing; F_DynBal was developed to contend with Duato's fully adaptive routing; and FD_DynBal was developed to contend with T3E-like routing.

The cost-effectiveness evaluation of each of the routing algorithms shows that:

1) Among the set of non-fully adaptive routers, DynBal is the best. Comparing to TRC, it has a slightly slower clock and a slightly higher cost, but much better performance than TRC because the virtual channel balance. Compared with Duato's partial adaptive routing algorithm, it has a slightly higher cost, the same clock rate, but much higher performance, especially when more chip area is available for alleviating the bottleneck and bubble problems. Compared with the T3D-like algorithm, it has a little cost-overhead if the LUT in T3D-like is shifted to the local node, but higher performance, especially under local traffic when T3D-like is optimized globally.

2) Between the two pure WH fully adaptive routers, F_DynBal is better. Although it has a slightly more cost overhead than Duato's fully adaptive router, it performs much better, especially when more chip area available, by alleviating the bottleneck and bubbles of the latter.

3) Between the two hybrid VCT and WH routers, FD_DynBal is better. The reason that it is more cost-effective than T3E-like is the same as DynBal is preferred over T3D-like.

The evaluation also shows that the DynBal algorithm is preferable under a random concentrated workload, no matter whether it is uniform, local, or non-uniform. Fully adaptive routing is preferable only for a non-random concentrated workload, either local or non-uniform. Among the fully adaptive routers, F_DynBal is most preferable not only because of its flexibility in a variable-packet-size system, but also because of its better performance under non-random local traffic and global non-random traffic with big packets. Even in the case where global non-random traffic is the only concern, FD_DynBal is a better design than T3E for cost-effectiveness, simple NIC, or simple packetizing process and flexibility.

5.9 A Plausible SoC Multicomputer

As presented in Section 1.2, a SoC multi-computer is plausible following the trends of process technology. Based on the application domain, the router network bandwidth can be either consumed by a single PE or by a complicated SMP system. Table 5.9 lists several of the processors that can be used as PE nodes – all the areas listed are based on the same ASIC technology as our router design, LSI G11-p.

In a .25 micron LSI G11-p ASIC technology, a fully adaptive router, F_DynBal, takes about 2.60 mm^2 of chip area. If EZ4102 is used as a local node PE, each node takes 6.18 mm^2. A 1-cm^2 die can hold as many as 16 nodes. The current maximum die size of LSI can reach 2.65 cm^2, which allows a 42-node multicomputer system on a chip. The chip has a maximum 1 MB on-chip SRAM, which can be used as cache in a COMA architecture – 23KB per node. It also allows as many as 1600 signal I/O pins so that each node can have 38 pins on average, which is enough for each node to access as big as several MB off-chip memory and some I/O ports.

Table 5.9. Plausible PE node in a SoC multi-computer.

IP Core	Source	Area (mm^2)	Features
EZ4102 TinyRISC	LSI Logic	3.58	R3000 MIPS-II, 16/32-bit intermixed compression, 3-stage pipelined CPU, MMU, Cache controller, basic BUS, Timer, UART
LeonSPARC	ESA	11.02 *	5-stage pipelined integer unit (32-bit SPARC V8 compatible), Cache/memory controller, UART, timer, PCI, AMBA internal bus
SIMD PE	CAAD/UH	.63	8-bit 4-stage pipelined PE used in SIMD

* -- not count I/D configurable cache.

More advanced process technologies are already commercially available (unfortunately, the more advanced technology than G11-p is not available for us), and is advancing rapidly. For example, a .18 micron LSI G12-p maximal die can have 33 million usable gates, four times more gates than G11-p, which means a 171-node SoC multicomputer. As the die size doubles every three year, the line width is halved

every seven years, and the logic speed doubled every three years, it is plausible that hundreds of more complicated nodes can be integrated on a single chip, with much faster (say, over 1 GHZ) routers. The scale of the SoC network will be probably be more constrained by the I/O pins or the on-chip memory. In this case, the UART of each node can be used to access off-chip memory serially. This is more possible in applications where the data domain is fixed and loading is rarely needed, for example, in some computer image processing applications.

6 DISCUSSION

6.1 Summary

The main goal of this dissertation is to evaluate the cost-effectiveness of router designs for the domain of torus buffered wormhole routing. This has been accomplished in a systematic manner. First, we presented a dynamic balanced scheme to enhance the performance and flexibility of previous algorithms. Second, we developed a flit-level, cycle-driven simulator for cycle-level performance evaluation. Last, we constructed routers based on state-of-the-art ASIC technology for detailed cost and speed evaluation.

The dynamically balanced routing algorithm is demonstrated to be free of deadlock by analyzing the extended channel dependency graph (after the introduction of a hardwired-dateline), and by illustrating the cases of packet-co-residence in the escape channel. The algorithm is shown to be more flexible than the statically-balanced scheme in the sense of both adaptability to workload and router implementation. It is also shown to enhance the performance of previous algorithms derived from the cycle-tolerance theory, and to extend their scope of applicability to buffered wormhole by removing both the restrictions on the escape channel and bottleneck formed there.

The simulator is built to model a canonical router parameterized by the design specifications of the networks, routers, workloads and routing algorithms. The performance criteria under dynamic workloads are expressed in flits per node per cycle for accepted traffic and in cycles to finish the transmission under static workloads. Some of the characteristics of the simulation system are high speed, user friendliness, flexibility, and fairness.

The routers implementing the various routing algorithms are constructed based on the same canonical model and the same state-of-the-art ASIC technology: this results in a fair comparison. Also, the use of a contemporary process technology sheds light on the influence of Moore's Law on router design tradeoffs. We can implement buffered wormhole algorithms as well as more complicated control as more gates are available, but also have a challenge in making the routers faster to meet the needs of higher speed nodes.

The results of the simulation and the router synthesis and timing analysis are combined to derive the cost-effectiveness evaluation of the design tradeoffs. The most detailed analysis is the comparison of the dynamically balanced routing algorithm devised here with its counterparts. Here are some conclusions of the cost-effectiveness evaluations:

1. The cost-effective routers (routing algorithms):
    - Constrained by a low to medium chip area budget and under a random dominated workload, no matter whether it is uniform, local or non-uniform, the DynBal router is the most flexible and cost-effective.
    - Constrained by a medium to high chip-area budget and under various workloads, F_DynBal is the most flexible and cost-effective.
    - With an unconstrained chip-area budget, FD_DynBal is preferable to T3E-like.

    Hence, we suggest that the DynBal be used as a low cost yet high performance router for a random dominated traffic and that the F_DynBal be used as a medium cost yet high performance router for a broader band traffic (or synthetic workload).
2. Other router design tradeoffs:
    - Improved process technologies will enable the use of adaptivity as it will decrease the impact of the cost and speed associated with complicated control circuits.
    - The delicate design of routing and arbitration logic can prevent them from being the critical parts of a fully adaptive router; thereafter the transmission data path becomes a primary concern. This also means that big switches formed by multiple lanes should be avoided in view of cost-effectiveness.
3. Taking into account the cost, rather than just the performance for evaluation dramatically changes the view of pros and cons of various design tradeoffs.
4. Other general lessons learned from this research are apparently the same as those discovered by the mainstream MPP community. The virtual channel or lane is expensive, for example, not only by the view of its chip area cost, but also in view of its slowed clock rate. As seen from the design of the F_DynBal router, the bigger switch size is one of the reasons for its lower performance than the TRC.

6.2 Contributions

The primary contribution of this work is the systematic study of cost-effectiveness of various router design tradeoffs. We have

1. Evaluated the design tradeoffs by considering cost and benefit, rather than just benefit (as has been done in most previous studies).
2. Compared the cost-benefit of adaptive routing with deterministic routing.
3. Presented a new dynamic balanced scheme and three routing algorithms derived from it (corresponding to three sets of routers):
   - A non-fully adaptive algorithm, DynBal, is more cost-effective than TRC, Duato's partial adaptive and T3D-like;
   - A fully adaptive WH algorithm, F_DynBal, is much more cost-effective than Duato's fully adaptive;
   - A fully adaptive (the hybrid of WH and VCT) algorithm, FD_DynBal, is both more cost-effective and flexible than T3E-like.
4. Developed a common platform consisting of two parts based on the same canonical model: One is the cycle-driven simulator, the other the router ASIC design module.

The approach itself, the tools developed and the results presented are also the contributions of this research.

6.3 Future Work

The first item would be to continue to use the platform to generate results: there are very many design tradeoffs and combinations that still need to be explored.

In the current simulator, a workload is generated according to the communication pattern specified and the mapping style – either dynamic or static. More realistic workloads are preferable. One way to accomplish this is to let a top level MPP simulator control the execution of the cycle-driven network simulator and measure the performance under a real workload environment established by the top level.

The F_DynBal router only takes a small part of the common chip area (1 cm^2), so it is possible to make a small-scale torus network on a single chip, as proposed in section 1.2. The chip can either act as a part of a clustered large MPP system or as an embedded co-processor in a PC or workstation. The processing element (PE) in this system can be either an IP core from vendors or a specially designed part.

BIBLIOGRAPHY

[1] Accelerated Strategic Computing Initiative (ASCI), ASCI Platforms, Lawrence Livermore National Laboratory, July, 2001, http://www.llnl.gov/asci/plateforms/

[2] Adve, V.S. and Vernon, M.K., Performance Analysis of Mesh Interconnection Networks with Deterministic Routing, IEEE Transactions on Parallel and Distributed Systems, Vol 5, No 3, March 1994, 225-246.

[3] Alverson, R., Callahan, D. and etc., The Tera Computer System, Tera Computer Company, 2000.

[4] Aoyama K., Design Issues in Implementing an Adaptive Router, MS thesis, Univ. of Illinois at Urbana-Champaign, 1993

[5] Aoyama, K. and Chien, A.A., The Cost of Adaptivity and Virtual Lanes in a Wormhole Router, Journal of VLSI Design, Technical Report, Dept. of CS, Univ. of Illinois at Urbana-Champaign, 1994.

[6] Anderson, T.E., Culler, D.E. and etc., A Case for NOW (Networks of Workstations), IEEE Micro, 1995, 54-64

[7] Athas, W.C. and Seitz, C.L., Multicomputers: Message-Passing Concurrent Computers, IEEE Transaction on Computers, Aug 1988, 9-24

[8] Berman, C.L. and Trevillyan, L.H., Global Flow Optimization in Automatic Logic Design, IEEE Transactions on Computer-Aided Design, Vol. 10, No. 5, May 1989, 557-564

[9] Bhatnagar, H., Advanced ASIC Chip Synthesis: Using Synopsys Design Compiler and PrimeTime, Kluwer Academic Publishers, Newport Beach, CA, 1999.

[10] Bolding, K. Non-uniformities introduced by virtual channel deadlock prevention. Tech. Rep. UW-CSE-92-07- 07, Dept. of Comp Sci. and Eng., U. of Washington, Seattle, WA 98195, 1992

[11] Bolding, K., Fulgham M. and etc., The Case for Chaotic Adaptive Routing, IEEE Transactions on Computers, Vol. 46, No. 12, Dec. 1997, 1281-1292

[12] Bolding, K., Cheung, S and etc., The Chaos Router Chip: Design and Implementation of an Adaptive Router, In Proceedings of VLSI '93, IFIP, 1993, 311-320

[13] Bolotski, M., Abacus: A Reconfigurable Bit-Parallel Architecture for Early Vision, Ph.D. Dissertation, Massachusetts Institute of Technology, 1996

[14] Boppana R.V. and Chalasani, S., A Comparision of Adaptive Wormhole Routing Algorithms, Proceedings of 20th Annual Intrnational Symposium on Computer Architecture, 1993, 351-360

[15] Chang, S., Ginneken, L.P. and Malgorzata, M., Circuit Optimization by Rewiring, IEEE Transactions on Computers, Vol.48, No. 9, Sept., 1999, 962-970.

[16] Chien, A. A., A Cost and Speed Model for k-ary n-cube Wormhole Routers, Proceedings of Hot Interconnects '93, Palo Alto, CA, Aug. 1993, 311-317.

[17] Chien, A. A. and Kim, J. H., Planar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors, Proceedings of the 19th International Symposium on Computer Architecture, IEEE Computer Society, May 1992, 268-277

[18] Compaq Computer Company, AlphaServer Technology, 2000

[19] Cypher, R. and Gravano, L., Storage-Efficient, Deadlock-Free Packet Routing Algorithms for Torus Network, IEEE Transactions on Computers, Vol 43, No 12, Dec. 1994, 1376-1385

[20] Dally, W. J. Virtual channel flow control. IEEE Trans. On Parallel and Distributed Systems Vol. 3, No. 2, March, 1992, 194-205

[21] Dally, W. J., and Seitz, C. L. The torus routing chip. Distributed Computing 1 (1986), 187-196

[22] Dally, W. J., Fiske, J. S. and etc., The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms, IEEE Micro, April 1992, 23-39

[23] Dally, W. J., Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels. IEEE Trans. On Parallel and Distributed Systems Vol.4, No.4, 1993, 466-475

[24] Dally, W.J., Dennison L.R. and etc., Architecture and Implementation of the Reliable Router, Proceedings of Hot Interconnects II, Palo Alto, CA, Aug. 1994, 122-133

[25] Duato. J., Why Commercial Multicomputers Do Not Use Adaptive Routing?, IEEE Technical Committee on Computer Architecture Newsletter, Summer-Fall 1994, 20-22

[26] Duato, J., Yalamanchili, S., and Ni, L. Interconnection Networks: An Engineering Approach. IEEE Computer Society Press, Los Alamitos, CA, 1997

[27] Duato, J., A Necessary and Sufficient Condition for Deadlock-free Adaptive Routing in Wormhole Networks, IEEE Trans. on Parallel and Distributed Systems, Vol.6, No.10, 1995, 1055-1067

[28] Duato, J., A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks, IEEE Trans. on Parallel and Distributed Systems, Vol.4, No.12, 1993, 1320-1331

[29] Duato, J., and Lopez, P. Highly Adaptive Wormhole Routing Algorithms for N-Dimensional Torus, In Proc. of DIMACS Workshop on Interconnection Networks and Mapping and Scheduling Parallel Computations, Piscataway, NJ, 1994

[30] Duato, J., and Lopes, P., Performance Evaluation of Adaptive Routing Algorithms for k-ary n-cubes, Proceedings of Parallel Computer Routing and Communication Workshop, May 1994, 45-59

[31] Felperin, S.A., Gravano, L. and etc., Routing Techniques for Massively Parallel Communication, Proceedings of IEEE, Vol 79, No 4, April 1991

[32] Fillo, M., Keekler, S.W. and Dally, W.J., The M-Machine Multicomputer, International Journal of Parallel Programming –Special Issue on Instruction-Level Parallel Processing Part II, Vol 25, No 3, 1997, 183-212

[33] Foster-Johnson, E., Graphical Applications with Tcl and Tk, 2e, M & T Books, IDG Books Worldwide, Inc., Foster City, CA, 1997

[34] Glass, C. J. and Ni, L. The turn model for Adaptive Routing, Proc. of the 19th International Symposium on Computer Architecture, May, 1992, 278-287.

[35] Hennessy, J. L. and Patterson, D. A., Computer Architecture: A Quantitative Approach, Morgan Kaufmann Publications, Inc., San Francisco, CA, 1996

[36] Herbordt, M. C., Ge, J., Sanikp, S., Olin, K., Le, H., Design Trade-Offs of Low-Cost Multicomputer Network Switches, Proceedings of the 7th Symposium on the Frontiers of Massively Parallel Computation, 1999, 25-34

[37] Hewlett-packard Company (HP), HP9000 Severs, 2000

[38] Horst, R.W., Tnet: A Reliable System Area Network, IEEE Micro, Feb. 1995, 37-45

[39] Horst, R.W. and Garcia, D., SeverNet SAN I/O Architecture, Hot Interconnects V, Stanford, CA, 1997, 1-8.

[40] Hwang, K., Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw-Hill, Inc., New York, NY, 1993

[41] Jump, J.R., Rice Parallel Processing Testbed, Fall 1993. http://www.rice.edu/softlib/rppt.html

[42] Katevenis, M., Serpanos, D. and Vatsolaki, P., ATLAS I: A General-Purpose, Single-Chip ATM Switch with Credit-Based Flow Control, IEEE Hot Interconnects IV Symposium Proceedings, Stanford, CA, August 1996, 1-11

[43] Katz, R.H., Contemporary Logic Design, The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1995

[44] Kim, J.H and Chien A.A., The impact of packetization in wormhole-routed networks, In Proceedinds of Parallel Architecture and Language Europe, June 1993, 242-253

[45] Lagman, A. and Najjar, W.A., Analysis of Buffer Design for Adaptive Routing in Direct Networks, Proceedings of the 4th Int'l Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1996, 134-140.

[46] Leiserson, C.E., Abuhamdeh, Z.S. and etc., The Network Architecture of Connection Machine CM-5, Journal of Parallel and Distributed Computing, Vol 33, No 2, 1996, 145-158

[47] Lockhart, J.B., JANET A Flit-Level Multiprocessor Network Simulator, July 1998. http://www.johnlockhart.com/research/janet

[48] Lutz, S., Manohar, S. and etc. RS/6000 7025 Model F80 Technical Overview and Introduction, International Business Machines Corporation (IBM), 2000

[49] LSI Logic Corporation, Using LSI Logic and Synopsys Design Tools and Libraries, Oct. 1998

[50] LSI Logic Corporation, G12 Cell-Based ASIC Library Release Notes, Oct. 1999

[51] McKeown, N., The iSLIP Scheduling Algorithm for Input-Queued Switches, IEEE/ACM Transactions on Networking, Vol 7, No 2, April 1999, 188-201

[52] Mattson, T.G. and Henry, G., An Overview of the Intel TFLOPS Supercomputer, Intel Technology Journal, Vol. 1, 1998, 1-19.

[53] Miller, D. and Najjar, W., Preliminary Evaluation of a Hybrid Deterministic / Adaptive Router, Proceedings of the Parallel Computing, Routing and Communication Workshop, Atlanta, June 1997, 1-10.

[54] Nesson, T. and Johnsson, S.L., ROMM Routing on Mesh and Torus Networks, Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures, July 1995, 275-287.

[55] Nguyen, T.D. and Snyder, L., Performance Analysis of a Minimal Adaptive Router, Proceedings of the 1994 Parallel Computer Routing and Communication Workshop, May 1994, 31-34

[56] Olin, K., Design Tradeoffs of Low-cost Multi-computer Networks, Master Thesis, Dept. of ECE, Univ. of Houston, 1999.

[57] Ould-Khaoua, M., A Performance Model for Duato's Fully Adaptive Routing Algorithm in k-Ary n-Cubes, IEEE Transactions on Computers, Vol. 48, No. 12, Dec. 1999, 1297-1304.

[58] Palnitkar, S., Verilog HDL: A Guide to Digital Design and Synthesis, SunSoft Press, Palo Alto, CA, 1996

[59] Park, J., Vassiliadis, S. and etc., Flexible Oblivious Router Architecture, IBM J. RES. DEVELOP. Vol 39, No 3, May 1995

[60] Pertel, M. J., A Simple Simulator for Multicomputer Routing Networks, Caltech Library System, Caltech-CS-TR-92-04, 1992

[61] Pinkston, T.M., Flexible and Efficient Routing Based on Progressive Deadlock Recovery, IEEE Transactions on Conmputers, Vol. 48, No. 7, July 1999, 649-669.

[62] Rexford, J., Feng, W., Dolter, J. and Shin, K., PP-MESS-SIM: A Simulator for Evaluating Multicomputer Interconnection Networks, Proceedings of the Simulation Symposium, April, 1995, 84-93

[63] Scott, S., and Thorson, G. Optimized routing in the Cray T3D. In Proc. Of the Workshop on Parallel Computer Routing and Communication (1994), 281-294

[64] Scott, S., and Thorson, G. The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus, Proc. of HOT Interconnects Symposium IV, Stanford U., Aug. 1996, 147-156.

[65] Scott, S. and Goodman, J., The Impact of Pipelined Channels on k-ary n-Cube Networks, IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 1, January 1994, 2-16.

[66] Schwiebert L. and Jayasimaha, D. N., Optimal Fully Adaptive Minimal Wormhole Routing for Meshes, Journal of Parallel and Distributed Computing, Vol.32, No. 1, Jan 1996, 103-117

[67] Shin, K. G., Chou, C. C. and Kweon, S. K., Distributed Route Selection for Establishing Real-Time Channels, IEEE Transactions on Parallel and Distributed Systems, March 2000, 318-335.

[68] Silicon Graphics, Inc. (SGI), SGI 2000 Series, 2000

[69] Steen, A.J. and Dongarra, J.J., Overview of Recent Supercomputers, 2000, http://www.phys.uu.nl/~steen/web00/overview00.html

[70] Stroustrup, B., The C++ Programming Language, Addison-Wesley, 1991

[71] Stunkel, C. B., Shea, D.G. and etc., The SP1 High-Performance Switch, in Proceedings of the Scalable High Performance Computing Conference, May, 1994, 150-157

[72] Stunkel, C. B., Shea, D.G. and etc., The SP2 High-Performance Switch, IBM System Journal, Vol. 34, No. 2, 1995, 185-204

[73] Stunkel, C. B., Challenges in the Design of Contemporary Routers, in *Lecture Notes in Computer Science*, vol. 1417, 1998, 139-152 (invited paper, Proc. of Parallel Computer Routing and Communication Workshop, Atlanta, GA, June 1997).

[74] Sun Microsystems, Sun and High Performance Computing, 2000

[75] Synopsys, Inc., Design Compiler Reference Manual: Fundamentals, 1997

[76] Synopsys, Inc., Design Compiler Reference Manual: Optimization and Timing Analysis, 1997

[77] Synopsys, Inc., Design Compiler Reference Manual: Constraints and Timing, 1997

[78] Tamir, Y. and Frazier, G. L., Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches, IEEE Transactions on Computers, Vol. 41, No. 16, Jun 1992, 725-737

[79] Tamir, Y. and Frazier, G. L., Hardware Support for High-Priority Traffic in VLSI Communication Switches, Journal of Parallel and Distributed Computing, Vol 14, 1992, 402-416

[80] Upadhyay J. H., Varavithya, V., and Mohapatra, P. A Traffic-Balanced Adaptive Wormhole Routing Scheme for Two-Dimensional Meshes, IEEE Transaction on Computers, Vol 46, No 2, Feb. 1997, 190-197.

[81] Washington University in St. Louis, The Interconnection Network Simulator (ICNS), 2000

[82] Zhang, X., Dasdan, A. Schulz, M and etc., Architecture Adaptation for Application-Specific Locality Optimizations, Proceedings of the IEEE International Conference on Computer Design, Austin, TX, October 1997, 150-156.