B-RPM: AN EFFICIENT ONE-TO-MANY COMMUNICATION FRAMEWORK FOR ON-CHIP NETWORKS

A Thesis

by

NOMAN SHAUKAT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2012

Major Subject: Computer Engineering

B-RPM: AN EFFICIENT ONE-TO-MANY COMMUNICATION FRAMEWORK FOR ON-CHIP NETWORKS

A Thesis

by

NOMAN SHAUKAT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Eun Jung Kim
Committee Members,	Rabi N. Mahapatra
	Paul V. Gratz
Head of Department,	Duncan M. H. Walker

August 2012

Major Subject: Computer Engineering

ABSTRACT

B-RPM: An Efficient One-to-Many Communication Framework

for On-Chip Networks. (August 2012)

Noman Shaukat, B.E., NED University of Engineering and Technology

Chair of Advisory Committee: Dr. Eun Jung Kim

The prevalence of multicore architectures has accentuated the need for scalable on-chip communication media. Various parallel applications and programming paradigms use a mix of unicast (one-to-one) and multicast (one-to-many) to maintain data coherence and consistency. Providing efficient support for these communication patterns becomes a critical design point for on-chip networks (OCN). High performance on-chip networks design advocates balanced traffic across the whole network, which makes adaptive routing appealing. Adaptive routing explores the path diversity of the network, increases throughput, and reduces network latency compared with oblivious routing.

In this work, we propose an adaptive multicast routing, Balanced Recursive Partitioning Multicast (B-RPM), to achieve balanced one-to-many on-chip communication. The algorithm derives its functionality from previously proposed algorithm Recursive Partitioning Multicast (RPM). Unlike RPM which uses fixed set of directional priorities and position of destination nodes, B-RPM replicates packet based on the local congestion information and position of destination nodes with respect to

current node. B-RPM employs a new deadlock avoidance technique Dynamically Sized Virtual Networks (DSVN). Built upon the traditional virtual networks, DSVN dynamically allocates the network resources to different VNs according to the run-time traffic status, which delivers better resources utilization. We also propose a new scheme for representing multiple destinations in packet head. The scheme works simply by differentiating multicast and unicast packets. The algorithm combined with dynamically sized virtual networks enables us to improve network performance at high load on average by 20% (up to 50%) and saturation throughput of network on average by 10% (up to 18%) over the most recent multicast algorithm. Also the new header representation scheme enables us to save 24% of dynamic link power.

To My Family

ACKNOWLEDGEMENTS

First and foremost, I will take this opportunity to thank my advisor, Dr. Eun Jung Kim who during the course of my research had been a source of guidance and motivation. I have greatly benefited from her advice and experience. I would also like to thank my committee members, Dr. Paul Gratz and Dr. Rabi Mahapatra, who provided me with their valuable advice and feedback on my research. I also thank Dr. Yum for helping me in improving my writing and presentation skills.

I am also grateful to all my High Performance Computing group colleagues, especially Lei Wang, Minseon Ahn, Hyunjun Jang, Baik Song An, Jagadish Chandar and Rahul Boyapati. I appreciate their time and effort which helped me in getting over the learning curve quickly and making my transition into group a smooth one.

Finally and most importantly, I would like to thank my parents who dedicated their lives for my future, who believed in me and always backed me throughout my life.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

An ever continuing process of CMOS feature size reduction has enabled us to put many processor cores in to a single chip [1]. Soon it will be very common for a chip multiprocessor (CMP) to have hundreds of cores within a single silicon die. Already we have processors like Intel 80-core Teraflop [2] and Tilera 64-core [3] that falls in this category**.** Providing efficient communication in a single die is becoming a critical factor for high performance Chip Multi-Processors. Traditionally shared buses had been used to support communication amongst the cores in CMP. But it is not hard to think why shared bus based network would fail when it comes to a large number of communicating entities. The number of cores in single die has surpassed the maximum limit of what a shared bus can handle. Winning the case over bus based networks, on-chip networks (OCN) have become an inevitable choice for many-core chip multiprocessors. Today we see chips like Intel 80-core Teraflop [2], Tilera 64-core [3] and TRIPS [4] that uses OCN as their communication architecture.

Unlike off-chip networks, on-chip networks have resource scarcity. They have limited area which limits the number of buffers to hold packets and they have power constraints. For Example, on-chip network router for 5 x 5 mesh operand network of TRIPS [5] occupies 10% of tile area. Similarly for power, Teraflop [2] reports around

---

This thesis follows the style of *IEEE Transaction on Very Large Scale Integration Systems.*

28% of tile power being consumed by the OCN, whereas in 16 tile RAW chip [6] the power consumption ratio jumps up to 36%.

Despite the scarcity of resources, significant research in this area has led us to OCNs that have fairly low communication latency and high throughput. However most of the research was on providing low communication latency or high throughput solutions for unicast traffic only. In order to achieve the near ideal performance, one should design the OCN to handle most of the communication primitives offered by vast variety of diverse application domains. One such type of communication involves sending the same message from one source to many destinations. Various parallel applications and programming paradigms use a mix of unicast (one-to-one) and multicast (one-to-many) to maintain data coherence and consistency. For example, Cache coherent shared memory system [7] relies on multicast messaging in order to function correctly. Similarly Token Coherence protocol [8] works its way out using broadcast messaging.

With lack of proper support for multicast messaging at OCN level, multicast messaging can increases the network latency very sharply. Same is true for network throughput; as small number of multicast packets can have a significant impact (negative) on network throughput [9]. As shown in figure 1, in a 4 x 4 mesh network, 1% of multicast load can cause the network to saturate at 25 % of link capacity instead of 45% of link capacity when we have unicast packet only [9]. The situation gets worse when the ratio of multicast packet increases. For example with 10% of multicast traffic,

network saturates at 15% of link capacity. Moreover lack of proper support for multicast traffic also results in high power consumption.

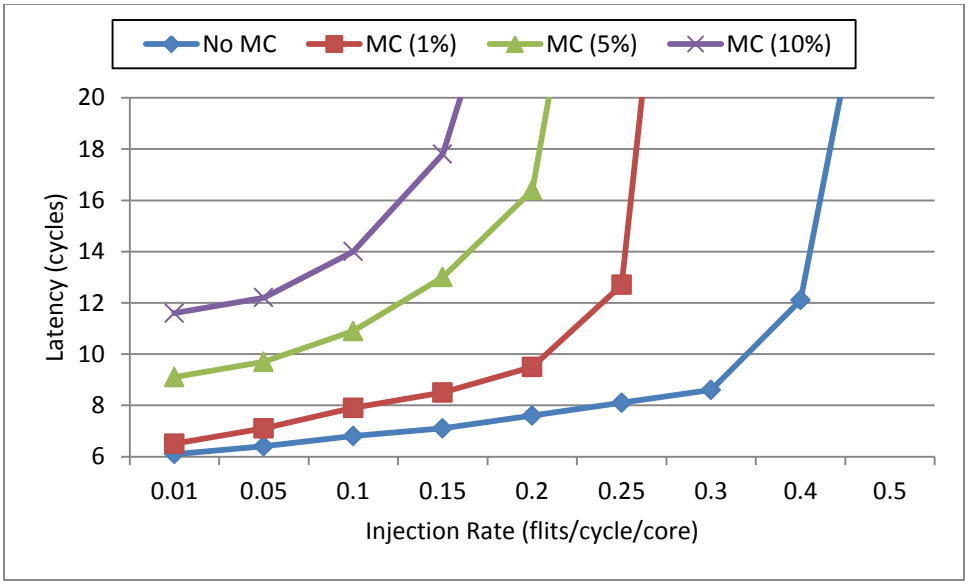


Figure 1. Network performance with varying ratio of multicast traffic

The reason for high communication latency and low throughput is that multicast packets generate a lot of unicast packets while traversing the network. If these packets are generated closer to source rather than destination they increase the load on network to a great extent and thus choking the network at very low offered load. A good multicast routing algorithm must delay the replication of packet as close to destinations as possible. If the packets are replicated at source then it is equivalent to multiple unicast messages. This means proper selection of replication points is instrumental in saving the bandwidth and link utilization. This also reduces the actual network load, improves network saturation throughput and save some power. For example, Figure 2 shows us the comparison on how the replication point decision can impact the performance. In Figure

2 node 9 wants to send a multicast packet to node 0, 1, 2 and 3. In Example 1 the multicast packet is replicated at source while in Example 2 replication was delayed until it was necessary to replicate. Though the number of replication operations is the same but Example 1 used 2.2 times more link bandwidth than Example 2.

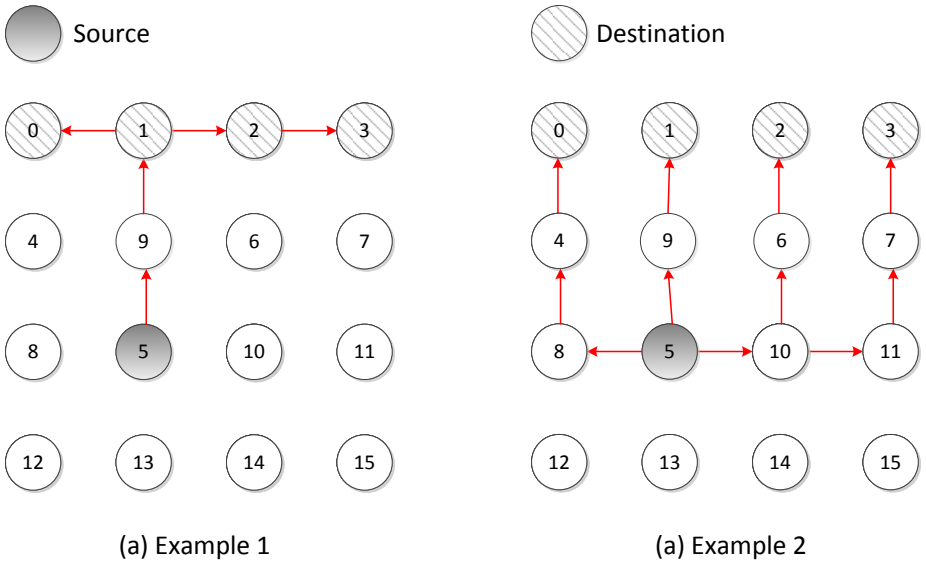


Figure 2. Link utilization for different replication schemes

In recent past there had been some research for efficient multicast support at OCN level [9-13]. One of the proposed algorithm, Recursive Partitioning Multicast [10] had been a leap over previous multicast algorithms for OCN [9, 11, 12]. RPM partitions the network into regions according to the current position of router and classifies the destinations of multicast packet into regions. Then according to position of active regions (regions with destinations) it computes the output path and makes the decision to replicate the packet or not. If the replication takes place than destination list are updated for each replicated packet accordingly. Though it showed remarkable improvement in

terms of performance over the previously proposed solutions, algorithm has some performance issues. RPM defines fixed priorities, has unbalanced resource allocation across vertical and horizontal ports and favors vertical ports over horizontal ports when it comes to broadcast or near broadcast messages.

Lately algorithms like BAM [13], MBR [14] and FANOUT [15] were introduced for multicast routing. FANOUT [15] is designed to cater broadcast or nearly broadcast messaging in OCN. Similarly BAM [13] and MBR [14] introduced adaptive routing for multicast but at the expense of extra resources.

In this research, we propose a multicast algorithm Balanced Recursive Partitioning Multicast (B-RPM) with new deadlock avoidance scheme. Local Congestion and positions of destinations is used by every intermediate router to make packet replication decision. Incorporation of Local congestion information helps in compensating the vertical port preference issue and buffer imbalance issue to a certain extent. To further offset the issue of buffer imbalance and provide deadlock avoidance, we introduce “Dynamically sized Virtual Networks”. DSVN allows us to utilize the network resources (i.e. buffers) efficiently and ensures an effective mechanism for deadlock avoidance. This work also introduces a destination list representation scheme built over previously proposed scheme [16]. The scheme exploits the fact that in general the number of unicast packets outnumbers multicast packets. Just by differentiating the two types of packets average header size can be reduced to great extent thus saving dynamic link power, the major source of power consumption in OCN. In general B-RPM reduce network latency by 20% on average (up to 50%) and defer network saturation by

10% on average (up to 18%) over the most recent multicast algorithm. Efficient resource utilization and small average size of header also saves us dynamic link power by 24%.

Rest of the thesis is structured as follows. Chapter II outlines the details of different components that constitute OCN. Chapter III presents background knowledge about multicast routing and discusses current algorithms used for multicast routing in OCN. Chapter IV provides the router architecture, multicast message structure and routing algorithm for B-RPM. Performance analysis and results for B-RPM are discussed in Chapter V and finally we present the conclusion of research in Chapter VI.

CHAPTER II

ON-CHIP NETWORK BUILDING BLOCKS

This chapter serves as the background for on-chip networks. In this chapter we will be discussing only those building blocks that are related to our research.

A. Topology

Topology can be defined as arrangement of communicating entities in the on-chip network. These arrangements vary from simple mesh, torus and ring networks to networks like butterfly. Figure 3 shows different topologies commonly used in OCN. Depending on the application needs or offered traffic pattern (and sometimes physical implementation feasibility) one topology may be preferred over another. Application needs can be quantified as the communication bandwidth they require, error handling capabilities and message latency requirement. These needs are mapped one-on-one to the measurable characteristics of topologies like bisection bandwidth, path diversities and average hop count. So the applications that demands path diversity will prefer kinds of mesh as their communication framework. The same goes with application where there is a chance to exploit locality. Application that demands uniform message latency across the network will prefer butterfly network.

2D mesh is the most commonly used topology in on-chip networks. 2D mesh offers simple design and regularity which make it easier to be laid out on a planner die.

It also offers path diversity which can be exploited to balance the load across the network [17]. Our algorithm is based on 2D Mesh, however with simple changes in algorithm it can be extended to higher dimension mesh networks.

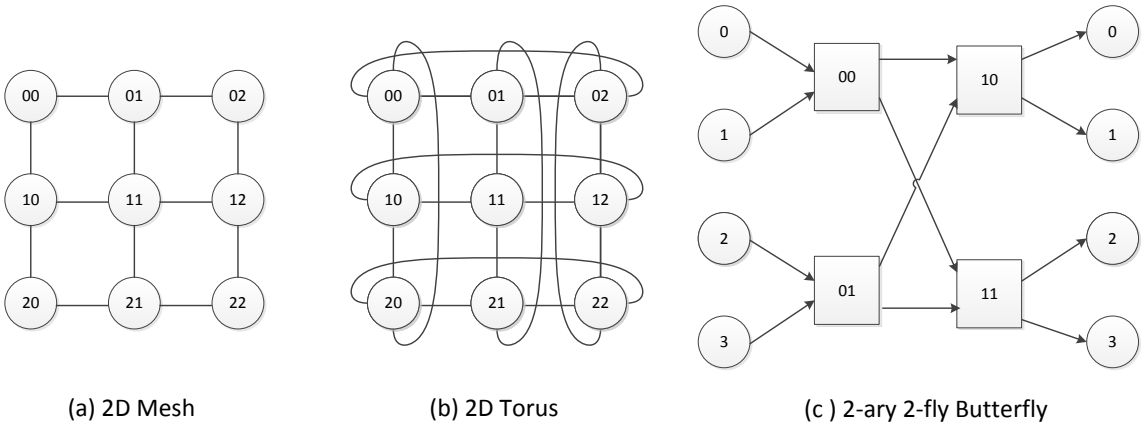


Figure 3. Different commonly used network topologies

B. Router Architecture

Router is the core component of on-chip network. Since our algorithm is based on 2D mesh, throughout this section we will be discussing the router architecture for 2D mesh.

1. Router Components

As shown in Figure 4, OCN router is composed to set of registers (that serve as buffers to hold flits, a small chunk of packet, while they wait for downstream router and

switch availability), an N x N crossbar switch to map input ports to output ports (where N is the number of input/output ports + injection/ejection ports. N in the case of 2D mesh is limited to 5) and a control logic that performs routing computation, handles flow control mechanism and does switch and buffer (virtual channel) allocation.

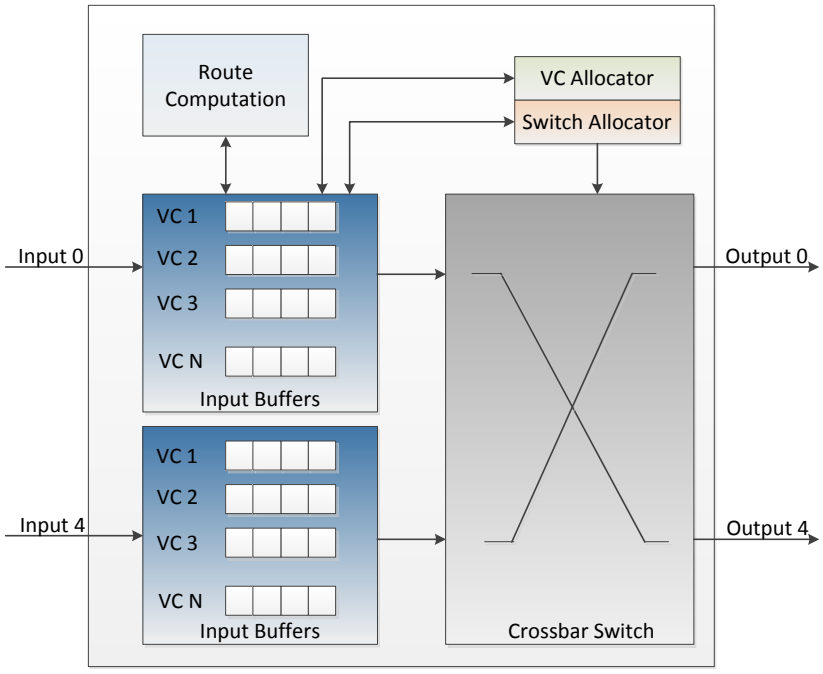


Figure 4. Baseline router architecture

Buffers in router are the main culprit when it comes to area and power consumption. In order to reduce the buffer requirement routers use wormhole-switching. In wormhole switching a packet is divided into number of flits. So the buffering is done at flit level rather than at packet level. Similarly components like Switch allocator also operates on flit rather than on packet level.

Router usually forwards the packet to one output port. Crossbar switch is the component that performs the actual forwarding. However in case of multicast one needs

the replication feature. Replication feature enables the router to replicate the same packet to multiple destinations. We need to modify the crossbar switch and the logic that allocates of crossbar switch to handle the mapping of more than one output port per input port. Control logic along with replication in router is discussed in detail in chapter III & IV.

2. Router Pipeline

Operation of router is pipelined to get higher network throughput. In general, OCN routers have 4 pipeline stages: Routing Computation (RC), Virtual Channel Allocation (VA), Switch Allocation (SA) and Switch Traversal (ST) [18]. One additional stage is required to transfer flit from one router to another. This stage is called Link traversal (LT), however this stage is not considered as router pipeline stage. Routing computation and Virtual channel allocation are unique to packet and thus only done once per packet. Whereas stages like Switch allocation and Switch Traversal are flit based. Since head flit contains information like destination list, head flit is responsible for allocation of resources like virtual channel and compute output port for other flits of the same packet. It has to pass through all 4 pipeline stages whereas body and tail flits need only the last 2 stages to get through the router. Tail flit; however has one additional responsibility that is deallocate the resources (i.e. buffers). The router pipeline is shown in Figure 5.

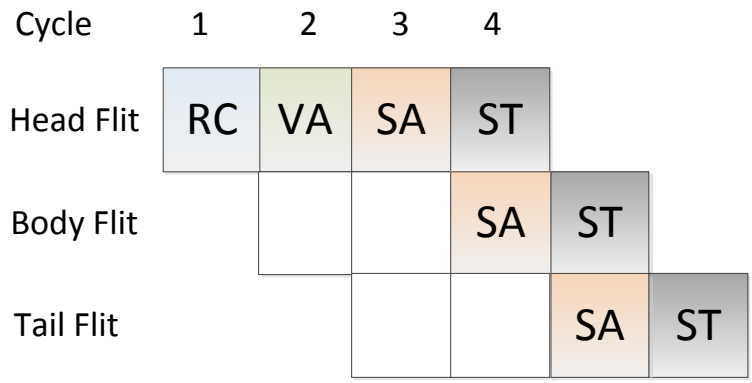


Figure 5. Router pipeline. (Color of each stage matches the components of router as shown Figure 4)

As soon as head flit arrives at router, RC stage computes the output port for the packet based on the destination information embedded in it. Meanwhile we may have the body flit arriving at the same router. Next step is to allocate a free virtual channel in the downstream router corresponding to the output port selected in RC stage. It is possible that there is no available virtual channel in the downstream router; in this case head flit has to wait for another cycle. This results in stalling of subsequent flits as well. Once VC is allocated, next step is to allocate switch. The first body flit at this time will enter VA stage (where it has to just wait for SA stage) and second body flit might enter RC stage (where it has to just wait for next stage). In SA stage, arbitration for input and output ports of crossbar is performed. If successful, head flit can use the switch in ST stage else flits of the packet have to stall again. SA and ST stages are no different for any flit. Every flit has to be allocated switch time before it can use the switch irrespective of the type of the flit. The next stage that usually is not the part of router pipeline is Link Traversal stage. In this stage a flit gets to the downstream router. Figure 6 shows the pipeline stages in case of a stall.

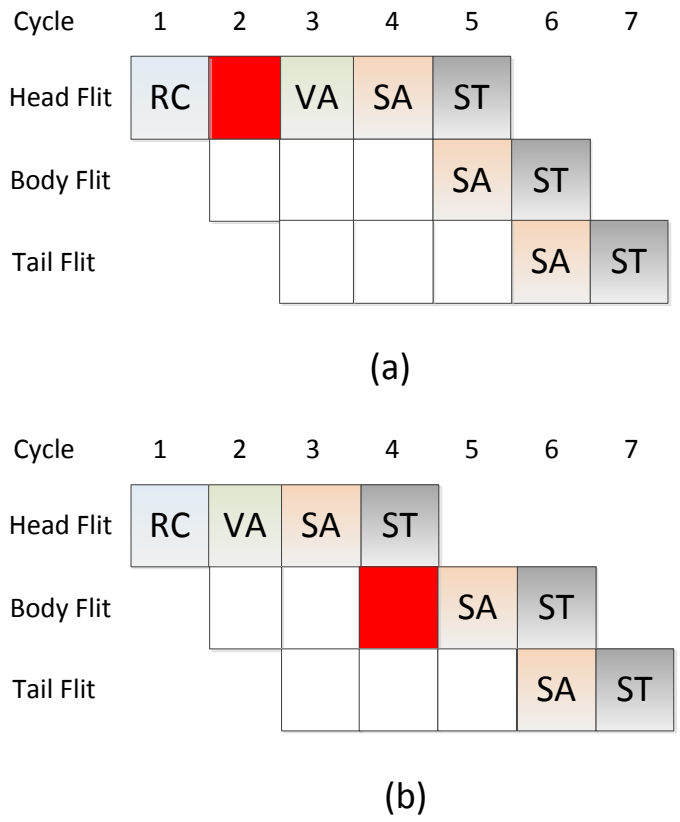


Figure 6. Router pipeline in case of stall. (a) Router pipeline in case of stall due to unavailability of virtual channel in downstream router (b) Router pipeline in case of stall due to failed attempt to allocate switch by body flit

Several techniques have been introduced to reduce the number of cycles a flit needs to traverse in a router. Some of the techniques that we use are mentioned below.

- **Lookahead routing:** In Lookahead routing, route computation for a packet is done a hop advance of the current router to eliminate the dependency between RC and VC stage of pipeline. The outcome is stored in the head flit. This way virtual channel allocation can be done as soon as the packet is received by the router.

- **Speculative switch allocation:** Sine there is no explicit dependency between VA stage and SA stage; SA stage can be done speculatively in the same cycle as VA [19]. If there is no free virtual channel then both VA and SA has to be performed again in the next cycle, whereas if SA fails then only SA has to be repeated again in next cycle. Figure 7 shows the effect of using lookahead routing and speculative switch allocation on router pipeline.

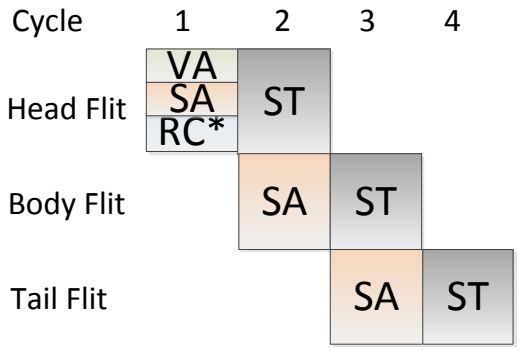


Figure 7. Router pipeline with lookahead routing and speculative switch allocation in place

C. Routing Algorithm

Another important component of on-chip networks is Routing algorithms. Routing algorithm defines a path for the packet from source to destination for a given topology. A well design routing algorithm should minimize the average hop length to destination as much as possible and try to uniformly balance the load in the network. Routing algorithms can be classified as following [20]:

- **Oblivious Routing:** In oblivious routing, router without considering the current state of the network, routes the packet towards destination. This also includes deterministic routing algorithm in which packet always take same path from source to destination. Oblivious routing algorithm can further be classified into minimal and non- minimal algorithm. In minimal oblivious algorithm the source router always select the shortest path to destination, whereas in non-minimal source router is free to choose any path towards destination.
- **Adaptive Routing:** In adaptive routing, router considers the current state of the network like network congestion while selecting path towards destination. Like oblivious algorithms it can be further classifies into minimal and non-minimal algorithm.

One important aspect of routing algorithm is how well it handles situations like deadlock and livelock. Deadlock occurs in the network because of the presence of cyclic dependencies amongst the resources. For example packets in Router A, B and C waiting for free buffers in Router B, C and A. In such scenarios packets are not able to move in the network and eventually whole network stalls. This concept is extended to 4 routers in Figure 8. Livelock on the other hand doesn't stall the network but in livelock packet never reaches destination and keep traversing the network.

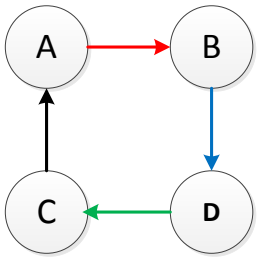


Figure 8. A deadlock situation in a 4-node network. Packets (coded in different colors) are waiting for resources in the downstream router

Algorithms can either choose to avoid deadlock at all or provide a sound recovery mechanism [17]. The algorithms that choose to avoid deadlock all together prevents the formation of cyclic dependency [21]. One approach to achieve deadlock avoidance includes restricted routing, that disallows the use of certain turns made by a packet enough to remove any cyclic dependencies between resources. The most common algorithm that follows this approach is Dimension order routing (DOR). For example, as show in Figure 9, in XY routing, a packet coded in red travelling vertically (in Y dimension) cannot make a horizontal turn (in X dimension). Whereas all the other packets coded in other colors (blue, green, black) are following legal path in XY routing.

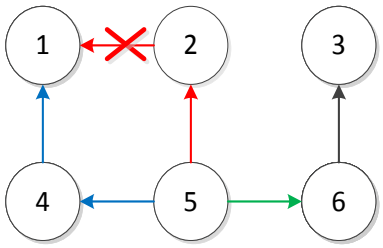


Figure 9. Turns allowed for packets in XY routing. (Packets are coded in different colors)

As the traffic increases in network, DOR approach may create hotspot in the network as they prefer some turns more than other. To avoid this situation usually OCN is divided into virtual networks with routing algorithms in those virtual networks complementing each other. Virtual Networks have been studied extensively in [22].

On the other hand the algorithms that prefer deadlock recovery can be of type regressive or progressive [17]. In regressive recovery packets involved in deadlock are removed from the network and source has to regenerate those packets wasting network bandwidth. In progressive recovery, algorithm resolves deadlock conditions without removing packet from network. Most common approach for progressive recovery is to use escape virtual channels [23].

CHAPTER III

MULTICAST ROUTING

As opposed to on-chip networks, multicast routing has been studied in off-chip networks for years [24-29]. Only recently researchers realized the need for multicast routing in OCN. They had to come up with some novel ideas in order to support multicasting in on-chip network because the stringent power and area requirements do not allow one to use the same off-chip ideas for OCN. This chapter briefly describes some of these multicast routing algorithms and highlights their performance issues. This chapter also outlines the applications and protocols that require multicast messaging.

A. Applications of Multicast Communication

With the advent of CMPs, several works have been proposed that either completely rely on multicast algorithm or their performance can be significantly improved if proper hardware support for multicast is present. In this section, we present some of the applications that use multicast communication.

1. Coherence Protocols

Multiprocessor chip needs some mechanism to maintain consistency amongst different copies of shared data in the cache local to each processing core in the chip.

Here Coherence Protocols come into play. They are the most common source of multicast traffic. Coherence protocols heavily rely on multicast messaging for their functionality. A variety of implementations have been proposed over time, some involving broadcasting the same message to all nodes in network and other being selective about their destinations. Some of the common implementations are discussed here:

- **Broadcast based designs:** In broadcast based designs a node in order to ensure coherence has to send message to every other node in the network. Broadcast based designs are very popular for ordering coherence request. In case of ordering request, shared bus has an advantage over OCN as it inherently maintains the order amongst the coherence requests. OCN usually offers path diversity which may lead to later requests reaching the destination nodes before the ones generated earlier. So message ordering has to be maintained explicitly. The Implementations like Token Coherence protocol [8], Intel Quickpath Interconnect protocol [30] and AMD Opteron protocol [31] uses broadcast messaging to achieve coherence request ordering. Uncorq [32] is another example that uses broadcast based design.
- **Multicast based Protocols:** Broadcast based designs are not scalable as it always assumes the every node had to be coherent with others irrespective of the actual need. In practice a node only needs to send coherence packet to a few other nodes. In large networks if broadcast based designs are used then it will

very likely saturate the network prematurely. Multicast based designs usually maintain a directory which tells the node, the destinations it needs to be coherent with. In some protocols directory node acts on behalf of request generator node and sends out multicast message to other nodes as in SGI-Orign [33] protocol. Other multicast based protocols include Multicast Snooping [34] and Destination Set Prediction [35].

2. Operand Networks Architecture

Operand networks communicate register values generated as result of some instruction to consumer tile which invokes the instruction waiting for result. In case of multiple instructions waiting on same result multicast messaging is required. TRIP [4] architecture is one example of Operand Network Architecture. Table 1 shows the percentage of multicast packets generated by some of the common protocols [9].

Table I. Percentage of multicast packets generated by common coherence protocols

Protocols	% of Multicast Packets
Directory Based Protocols	5.1
Token Coherence	5.5
Region-Based Coherence	8.5
Operand Networks	3.1

B. Multicast Routing Algorithm

A very basic multicast algorithm can be realized by sending out multiple unicast packets from the source node. Initially many proposed router designs had assumed that an efficient unicast mechanism can handle multicast or broadcast messaging by sending out multiple unicast messages [36-38]. The problem with such approach is the negative impact on network saturation. As shown in Figure 2, multiple unicast packets from same multicast packet waste useful bandwidth. The bandwidth could have been saved if a single packet would have travelled through the network until it was unavoidable to replicate. On the other extreme, we have techniques in which the same multicast packet visits each destination in its destination list one after another till the point where all destinations have been served. This approach is viable for very small networks. For large networks the destination in the last will suffer from high message latency. Obviously a better multicast algorithm should try to find a balance amongst these extremes. Number of solutions have been proposed that can be classified as a balance between the two extremes [10, 12, 14, 39-42], VCTM [9] being the first. These algorithms involve replicating packet in intermediate routers. A few are discussed below.

1. Virtual Circuit Tree Multicasting Algorithm

In VCTM, each multicast packet has to establish a tree shaped virtual circuit connection between a source node and destination nodes. Before sending out the actual

multicast packet, source node sends out setup packets with unique circuit id to each destination in multicast packet. While traversing these setup packets make an entry into a special table 'virtual circuit table', indexed by circuit id, maintained at each intermediate routers. Setup packets looks for the table entry at each intermediate router using circuit id. If entry is not found then a new entry is made into the table and output port for the setup packet is registered. If the entry is found and the current setup packet has a different output port from the output port registered in the entry, new output port is registered in the same entry and this router is designated as replication router. Next time if another packet is generated from the same source with same destination list, no setup is required for this multicast packet. The packet can use the same circuit id and can follow the circuit already established. The packet using virtual circuit id will pull the entry on every intermediate router and if more than one output ports are registered, router will replicate packets to all registered output ports.

VCTM [9] has some major performance issues. First and foremost setup packets waste a lot of network bandwidth. Setup packets are same as generating multiple unicast packets for a multicast packet. Other major issue is with the assumption that the circuit will be reused frequently. However it is highly unlikely that the same source node will be generating the multicast packet with same set of destinations. As the number of multicast to unicast packet ratio will increase setup packets will flood the network. Beside performance, VCTM [9] is expensive in terms of area as well. Virtual Circuit table occupies a lot of area in each router.

2. Recursive Partitioning Multicast Algorithm

RPM [10] is a leap over its predecessors. RPM proposed an elegant yet an efficient algorithm that eliminated the setup cost incurred in VCTM [9]. Based on the current location of packet, it partitions the network into at maximum 8 parts and replicates the packet to router's output ports depending on the position of partitions (with destinations) and some predefined priorities. It also proposed some optimized priorities that overrule the basic priorities in order to delay the replication as further as possible. We will be discussing RPM [10] in detail as it will serve as foundation for our work.

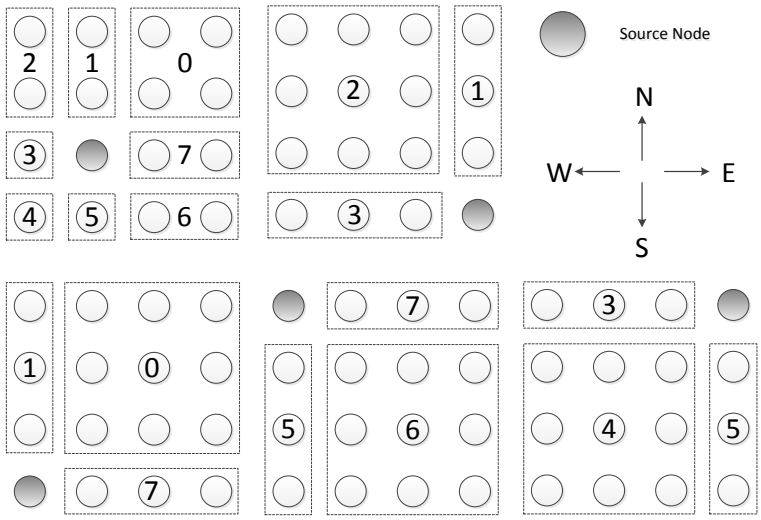


Figure 10. Possible network partitions based on source node positions

In RPM [10], when a router receives multicast packet it divides the network into at most eight regions depending on the position of routers. Some of the possible partitions are shown in Figure 10. In RC stage, the multicast router using the destination list computes output ports to which packet will be replicated. Output ports unlike unicast

routing may not be unique as position of destinations may be completely orthogonal. For destinations in partitions not diagonal to the current router the routing decision is simple but for destinations in diagonal partition, the algorithm has some fixed output port priorities and some optimization rules.

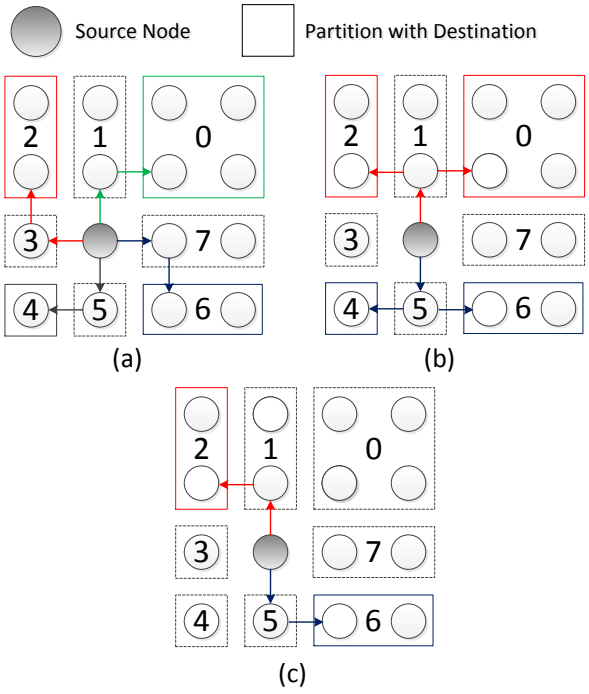


Figure 11. Priority rules for RPM. (Parts and corresponding packets and are coded in different colors)

- As shown in Figure 11(a), priorities are defined counter clock wise. For example North direction has priority over East direction for destinations in Partition 0, West direction has priority over North for Partition 2 and so on.
- If Partition 0 and 2 both has destinations then previous rule will be void and North port will be selected for destinations in both partitions. Same is the case for South port. This rule is illustrated in Figure 11(b)

- If Partition 3 does not have any destination but 1 and 2 have then North direction will be used instead of West. Again same rule applies for South direction. Figure 11(c) demonstrates the third rule.

For deadlock avoidance, RPM [10] used 2 Virtual Networks; each virtual network has imposed a turn restriction. VN 0 does not allow a packet to make a turn to south direction whereas VN 1 disallows a North turn. Configuration is visualized in Figure 12. The network in divided into 2 virtual networks Red and Blue. Red corresponds to VN 0 as it disallows the packet to move in south direction and Blue corresponds to VN 1 as it restricts north direction for the packet.

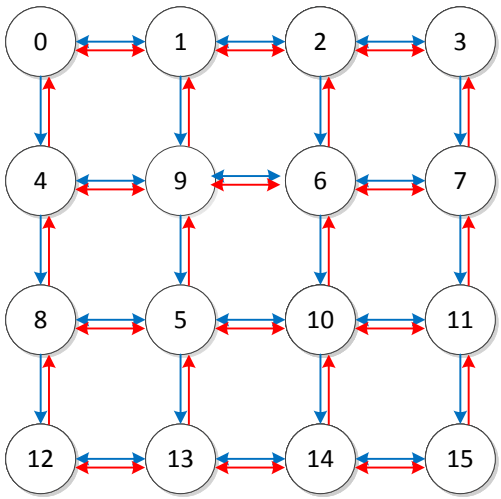


Figure 12. Virtual networks in RPM

RPM [10] has been a major improvement over previous multicast algorithms, but it has a few short comings. For example, there are twice as many buffers available in each virtual network in vertical direction than in horizontal direction. This can be easily

visualized in Figure 12, as we can see traffic for both virtual network in horizontal dimension whereas traffic for only one virtual network in vertical direction. This creates a severe imbalance of buffers across horizontal channels and vertical channels. Packets traversing through vertical channels are free to take any buffers as vertical channels exclusively belong to a single virtual network depending on the direction of packet traversal. But in case of horizontal channel resources have to be shared amongst virtual networks. This means in horizontal channels number of buffers in each VN is half of that in each VN in vertical channel.

Other issue in RPM comes from its optimization rule which states if packet has destinations in northeast and northwest partitions or southeast or southwest partitions, then basic dimension priority rules are overridden and north or south ports for both diagonal partitions take priorities respectively. For example if destinations are in partition 0 and 2 then in order to avoid a replication we can send a single packet to North instead of replicating the packet in west and north direction. For small destination list this seams a good move but as the destination list grows or the destinations in destination list starts getting sparse then the algorithm will always prefer vertical dimensions and replicating the packets in horizontal direction, as shown in Figure 13. This results in more horizontal link utilization than the vertical link utilization. The problem exaggerates as already there are less horizontal buffers available in each virtual network.

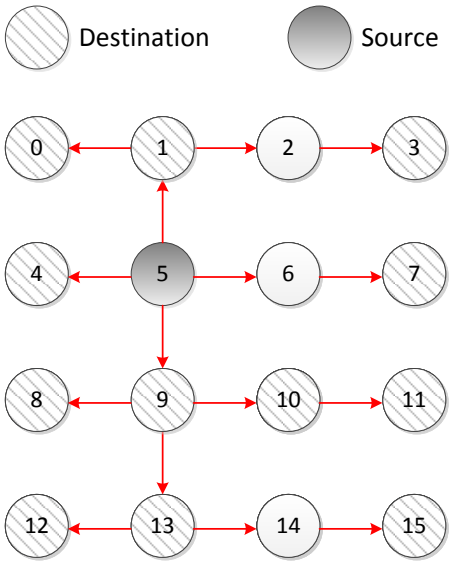


Figure 13. Excessive horizontal link utilization. (because of priority given to vertical ports)

3. Balanced Adaptive Multicast Algorithm

BAM [13] is an increment over RPM [10]. BAM uses the same concept as RPM [10], however BAM [13] instead of having fixed priority allows router to adaptively select the output port for destinations in diagonal partitions. Unlike RPM [10] which uses virtual networks to avoid deadlock, BAM [13], uses Duato's deadlock avoidance theory [43]. The algorithm allows packet to make any turn at intermediate routers adaptively without imposing any turn restriction. In case of any potential deadlock situation algorithm reroute the packet to escape virtual channel that implement deadlock free dimension order routing algorithm.

The escape channel scheme works efficiently if deadlock happens rarely. As the number of deadlock increasing, escape channels easily get congested and become the

bottleneck of the network. Figure 14 compares the percentage of packets that use escape channels between unicast and multicast traffic. It is observed that multicast traffic has higher percentage of packets using escape channels than unicast, which implies that in multicast traffic deadlock happens more frequently. The reason is that in multicast traffic a packet targets for multiple output ports of the same router, a packet will hold the router resources until all the replicas of the packet are sent to the requested output ports. It takes longer time for a packet to release the holding resources in multicast than in unicast. Networks with multicast traffic are much easier to fall into deadlock than those having only unicast traffic. Unless one is willing to add more escape channels, employing the escape channel scheme in multicast routing may not be an efficient design.

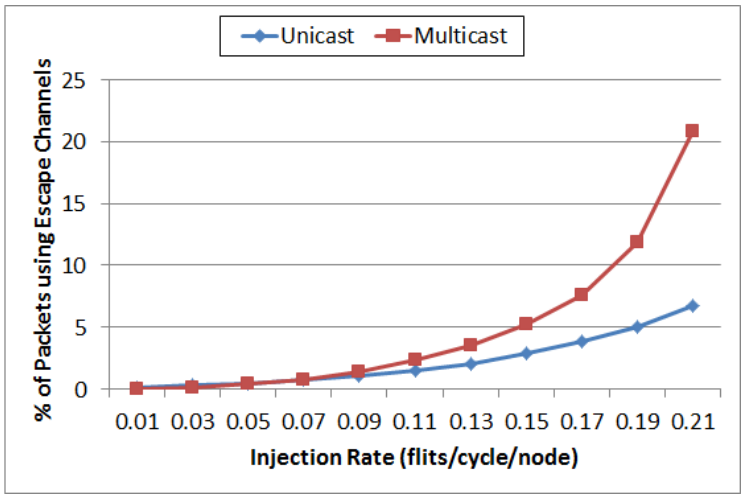


Figure 14. Percentage of packets using escape channels in unicast and multicast

Also BAM assumed the multicast packet to be of size one flit and made a case for smaller escape channels. However this assumption is not always valid, as in case of cache update protocols [44] multicast packet may contain more than one flit. Also escape channel implements either XY or YX routing which in case of multicast packets cannot accept all packets. For example, suppose escape channel implements XY routing, then north escape channels cannot allow any packet with destinations in northeast and northwest partitions, as they will violate XY routing.

So with BAM [13], one is stuck in area performance tradeoff. If one tries to maximize performance then routers need more buffer space for escape channels to drain as many packets as possible in case of potential deadlock. If one saves space then packets will experience high average network latency.

CHAPTER IV

BALANCED RECURSIVE PARTITIONING MULTICAST ROUTING

B-RPM uses adaptive routing and dynamically sized virtual networks to deliver packets to their destinations. This chapter gives detail description of the B-RPM routing algorithm and the infrastructure changes required to support the algorithm.

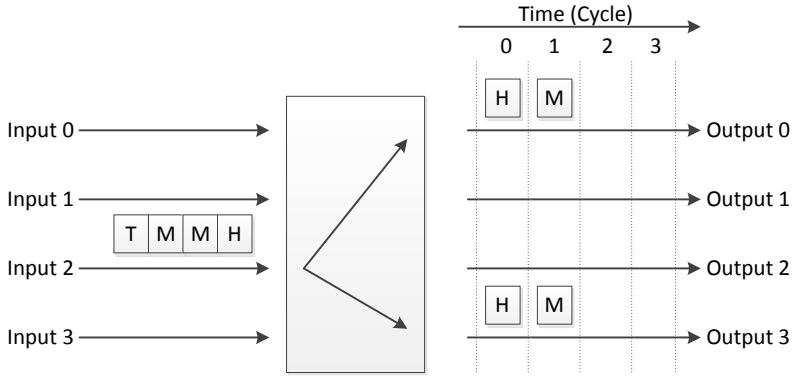
A. Router Architecture

B-RPM uses similar router architecture as discussed in Chapter II. However due to the characteristics of multicast traffic some changes need to be incorporated into a regular OCN router. For example, in order to replicate packet to downstream routers, proper replication support needs to be built into router. Crossbar switch besides one-to-one mapping between input and output should allow one-to-many mapping. Since the algorithm uses adaptive routing, pre-computing next hops is also a challenge and needs to be considered.

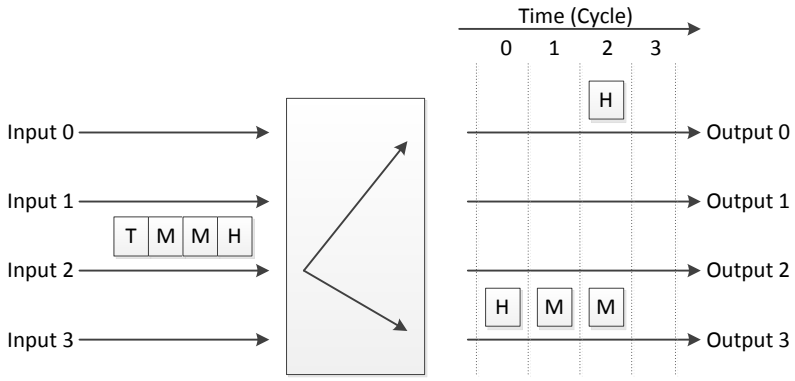
1. Replication Scheme

Replication, as proposed in [45], can be performed synchronously or a synchronously. As shown in Figure 15 (a), in synchronous replication, a flit is only forwarded to its output ports if all the output ports are available and flit can be forwarded

simultaneously. So if one port is not available the flit will not be forwarded to any of the output ports. This scheme results in high packet latency and high probability of deadlock. In asynchronous mode, replication to each output port can take place independently. This allows one to avoid deadlock situation and decreases the packet latency as flits are no more waiting for the availability of all port. Asynchronous scheme is visualized in Figure 15 (b). As in RPM [10], B-RPM also uses asynchronous replication.



(a) Synchronous Replication



(b) Asychronous Replication

Figure 15. Different packet replication schemes

2. Routing Computation Unit

Routing Computation Unit is responsible for computing the next hops for the packet. In B-RPM this computation is performed using the destination list embedded in packet header and local congestion information. With lookahead routing technique [18], B-RPM selects the best output port adaptively for each partition one cycle ahead [46-48]. This section omits the details on how B-RPM calculates the output ports as a detailed discussion is done later in this chapter.

3. Virtual Channel Allocator Design

With asynchronous replication, virtual channel allocation is not the same as for router that supports unicast packets only. Traditionally once tail flit of the packet gets into SA stage, current router VC is deallocated and is available for any packet in upstream router waiting for a free VC in the current router. However, in B-RPM a packet can be replicated to multiple output ports and the replication is done asynchronously. There can be case in which VC is not available for some downstream routers and the packet cannot be forwarded to that downstream router whereas for other downstream router packet can be forwarded. In this case, allocated VC in the current router cannot be deallocated until all replicated tail flit gets into SA stage.

4. Switch Allocator Design

The switch allocator is standard two stage arbiter [18]. First stage of arbitration selects a VC from each physical channel. The second stage groups VC according to their output port and selects one VC per output port. In multicast routing, a packet may need to replicate itself to different output ports. Now in second stage of arbitration we may have the same packet (VC) contending for different output ports. Since arbitration is performed in round robin manner there is a high chance that not all the requested output ports from the same VC are allocated at once. For ports that are allocated, a copy of flit is made and allowed to use the switch to be transferred to downstream router. For unsuccessful ports, the VC is queued again for arbitration.

B. Multicast Packet Header Structure

Multicast packets, by definition, are intended to be delivered to multiple destinations and destination information has to be embedded into head flit of packet. Several schemes have been proposed to encode multiple destination addresses into header. Some of them include multiple regions broadcast encoding, bit string encoding and all-destination encoding [49]. Each has its advantage and disadvantage. For example in multiple regions encoding, if a range of continuous address are found it is being replaced by first and last address. The scheme is good only for multicasts in which destinations are clustered. Similarly in destination id list encoding, individual addresses

of destinations are placed in header making header length very large if destination list is large. With Bit string encoding a bit vector of size equal to the number of nodes in network is embedded in header. Again since the most of the packets will not carry more than one destination, encoding the complete vector list makes no sense.

In order to minimize the header size, Lei et al. [16] proposed a header compression scheme for OCN with multicast packets. The scheme is based on the way partitioning is done in RPM. Since each output port can serve at most 3 partitions, one doesn't need to send complete bit vector in each direction. Information about those 3 partitions is sufficient to identify destinations. To further compress the bit vector, index bits for partitions are also embedded in the header along with a bit to identify if the header is compressed or not. An active bit in the index vector indicates that partition has at least one destination. If the bit is disabled, bit vector for that partition can be completely omitted. This scheme, though good at compressing header bits, is a designer's nightmare. With each router having different number of partitions and partition sizes, one can imagine the number of different circuits needed to be designed and tested for this scheme. Figure 16 shows commonly used header structures.

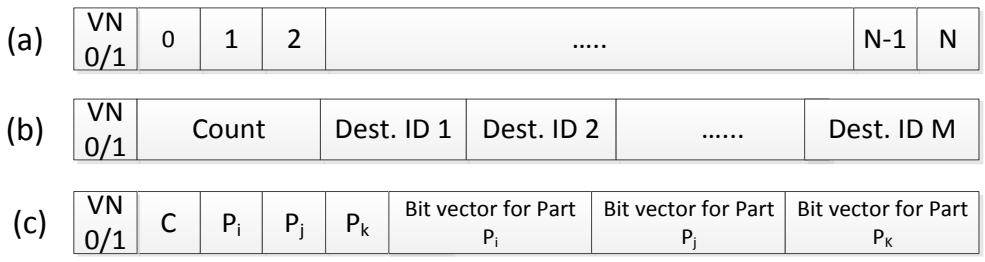


Figure 16. Different header formats. (a) Bit vector (b) Destination ID list (c) Compressed header format

B-RPM uses a variation of bit string encoding. As shown in Table 1, since most of the packets in the network are usually unicast, encoding complete bit string or even the compressed header will lengthen the packet header. One must distinguish between unicast and multicast packets. A bit in the header can be used to identify if the packet is unicast or multicast. If the packet is unicast, then destination identifier will be embedded into header instead of bit vector (this scheme treats all the packets with one destination in their headers as unicast packets. This is also true for the packets which are offshoots of Multicast packets).Whereas for multicast packets, one can use bit vector or compressed header if one is willing to afford design time complexity. Figure 17 shows the proposed header format.

Figure 17. Proposed header format. 3rd bit is used to differentiate a unicast packet with multicast packet.

Example of header compression along with the comparison with other commonly used header format is shown in Figure 18 and Figure 19. In a 4 x 4 network, initially node 9 wants to send packet to destination nodes 6, 10 and 11. It generates a single multicast packet and forwards the packet to east output port. Figure 18 shows the comparison amongst the commonly used header formats and the proposed format.

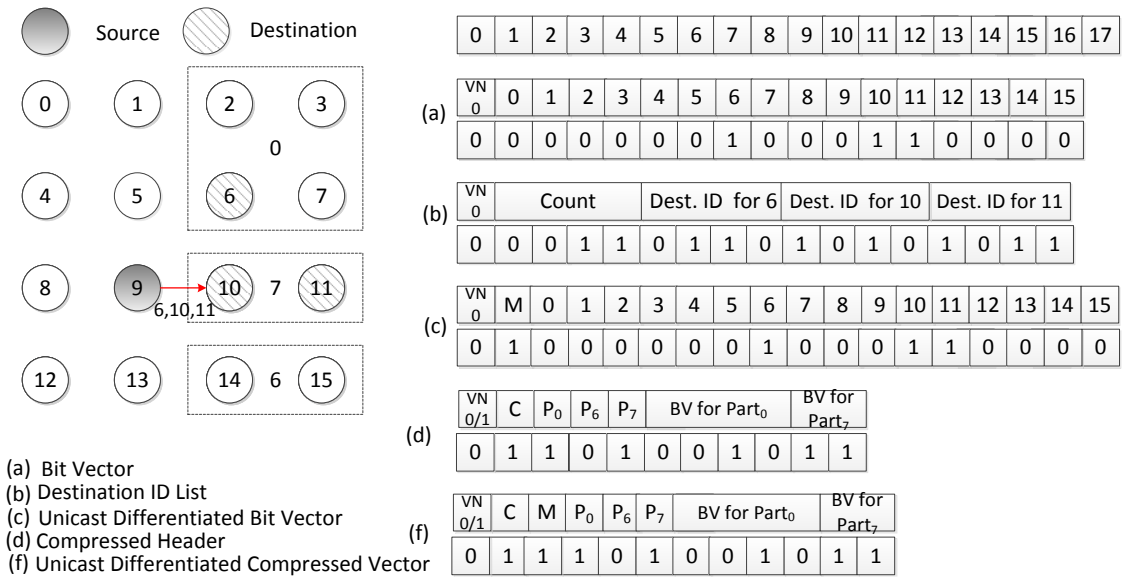


Figure 18. Comparison of different header formats in case of multicast packet

For unicast packet, Let say node 9 wants to send packet to destination node 6. It selects the east output port again. Figure 19 shows the headers in case of unicast packets.

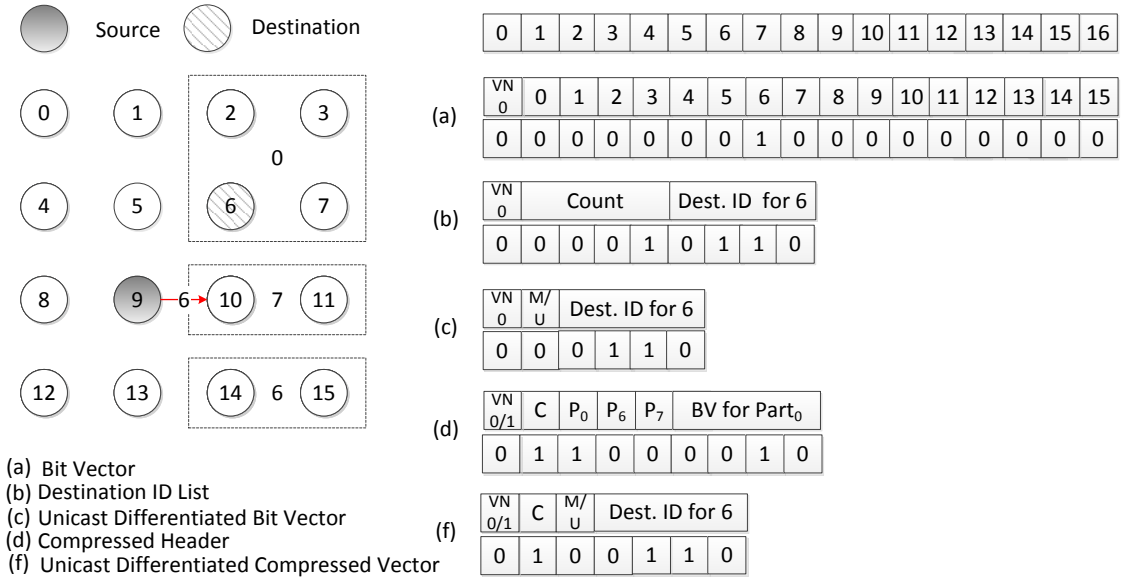


Figure 19. Comparison of different header formats in case of unicast packet

Please note. We have one extra bit in packet header. This is to identify the virtual networks. Virtual Networks are explained later in this chapter.

C. B-RPM Routing Algorithm

Routing decision in B-RPM is based on the position of destination nodes with respect to current router and local congestion information. Following the idea proposed in RPM [10], current router classifies the position of destinations into at maximum 8 partitions depending on the position of current router, as shown in Figure 10. When a router receives packet either via injection port or upstream router, it identifies the partitions where the destination nodes are located. Then using active partition list and local congestion information it computes whether the packet will be replicated or just forwarded and to which ports it will be replicated or forwarded. This is all done in RC stage.

Unlike unicast packets where there is only one port where the packet can be forwarded, a multicast packet can be replicated to multiple ports with each replicated packet having a filtered destination list according to their output port. For example if source node, replicates a packet in north and south direction then destination list of packet replicated to south will not have any destination located north to current router and destination list of packet replicated to north will not have any destination located south to current router. This is to ensure that destination nodes do not receive the same copy of packets more than once.

B-RPM uses minimal adaptive routing and routing decision is mainly based on the local congestion information. So, let say, if a destination is in partition 0 than source node can choose either north or east direction based on the local congestion situation. For destination is in Partition 1, since it is a minimal adaptive algorithm it had no choice but to go for north direction. Following are the rules for routing under B-RPM (also illustrated in Figure 20),

- For destinations exactly Vertical or Horizontal to current router, output ports in corresponding directions are chosen. For example destinations in Part 1 will be catered by packet replicated using north port. The rule is illustrated in Figure 20(a).
- As shown in Figure 20(b), for destinations in diagonal partitions (which can be reached from two directions in case of minimal routing) priority will be given to the direction with less local congestion unlike that of RPM which has fixed Priorities. For example for Part 2 if the west link is more congested then north link then north port will be preferred to cater destinations in Part 2. In case of equal congestion, vertical directions will be preferred.
- Priority for diagonal partitions based on local congestion will be overruled if the other direction valid for that diagonal partition is enabled due to direct Horizontal or Vertical destinations and the direction with less congestion is enabled only due to diagonal partition. For example, if west link is less congested than north link

and Part 1 and Part 2 have destinations than north link will be selected for Part 2 rather than west link. This rule is visualized in Figure 20(c).

- If east or west direction are chosen over North or South for both the diagonal partitions at their ends, (for example if east port is chosen for destinations both in part 0 and part 6) then only one partition will be served by east or west direction and one had to be transferred to north or south direction depending on which direction is less congested. This rule is imposed because virtual networks cannot allow a packet to simultaneously carry destinations that are both in North and South directions. This rule is show in Figure 20(d).

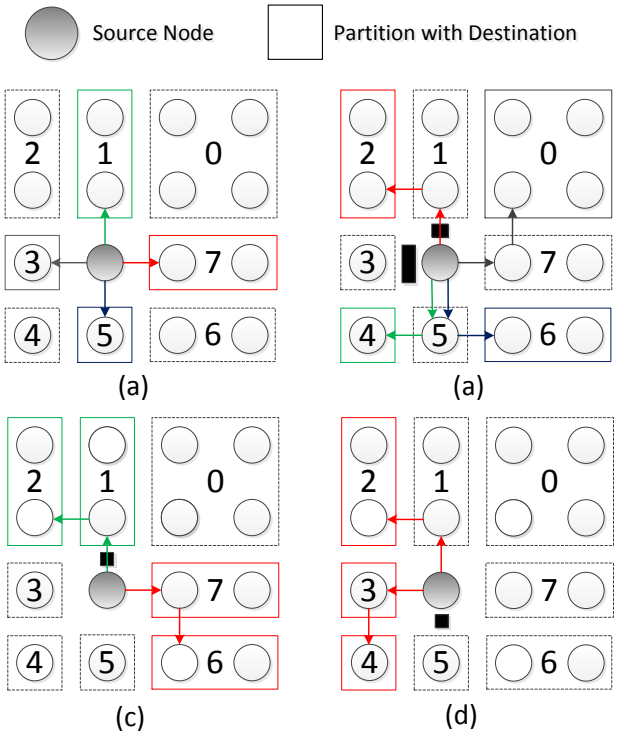


Figure 20. Priority rules for B-RPM. (Parts and corresponding packets and are coded in different colors and size of solid black box indicates link congestion)

B-RPM algorithm can be divided into two phases. In the first phase destination list is being parsed to find out the active partitions. As mentioned in packet header structure, we can have either bit vectors or a destination ID depending on whether the packet is multicast or unicast. Parsing is done accordingly. In the next phase using the active partitions and local congestion information output ports are being computed. Pseudo code for output port selection is show in Table II, whereas the hardware implementation of output port selector is presented in Figure 21.

Table II. Pseudo code for output port selection

```
W = In(dest,3) OR
      (In(dest,2) AND Conf(2,W,N)) OR
      (In(dest,4) AND Conf(4,W,S))

E = In(dest,7) OR
     (In(dest,0) AND Conf(0,E,N)) OR
     (In(dest,6) AND Conf(6,E,S))

N = In(dest,1) OR
     (In(dest,0) AND Conf(0,N,E))  OR
     (In(dest,2) AND Conf(2,N,W)) OR
     (Conf(*,N,S) AND (
           (Conf(2,W,N) AND Conf(4,W,S)) OR
           (Conf(0,E,N) AND Conf(6,E,S))
     ))

S = In(dest,5) OR
    (In(dest,4) AND Conf(4,S,W))  OR
    (In(dest,6) AND Conf(6,S,E)) OR
    (Conf(*,S,N) AND (
       (Conf(2,W,N) AND Conf(4,W,S)) OR
       (Conf(0,E,N) AND Conf(6,E,S))
    ))
```

In (destination_List Dl, Partition P): In function returns true if any member of DL lies in Partition P.

Conf (Partition P, Output_Port OP1, Output_port OP2): Conflict function returns true for a particular output port OP1 if it is a likely choice over other output port OP2 for partition P. Conflict arises between OP1 and OP2, if both parts directly corresponding to these output ports does not have destinations. The likelihood decision is then based on the local congestion in each output port.

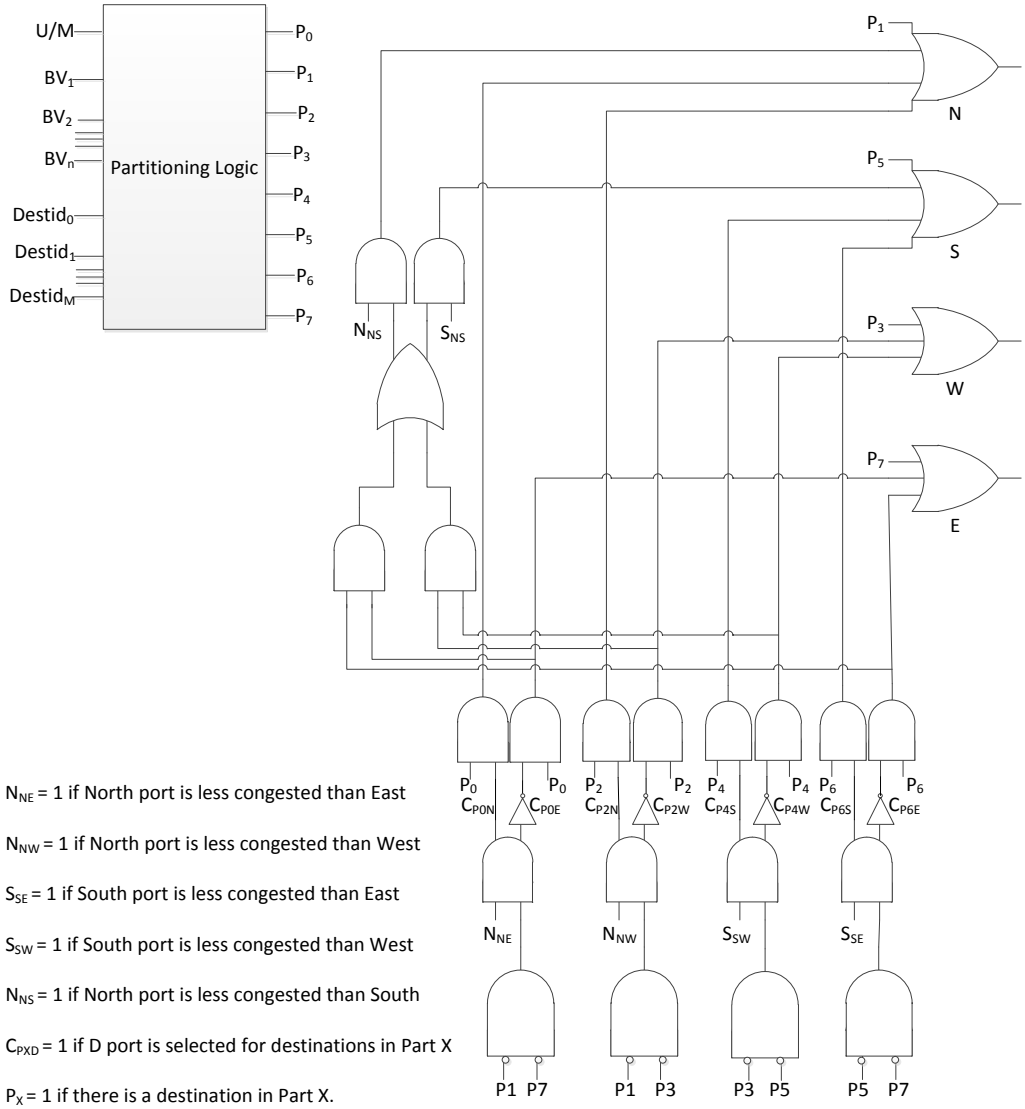


Figure 21. Hardware implementation of output port selector

To clarify the rules, we walk through an example as shown in Figure 22. Let's begin with zero load network. Let's say Node at 9 injects one multicast packet in to the network, router at Node 9 in routing computation stage parses the destination list in the packet head and let's assume the destination nodes are 0, 2, 3, 13, and 15. According to Node 9's position within network, the destination nodes lie in four partitions (0, 2, 5, and 6). Based on the routing rules, for destinations 13 and 15, Node 9 will replicate packet in south direction. As there is no destination in partition 7, south direction wins the right for sending the packet to node 15.

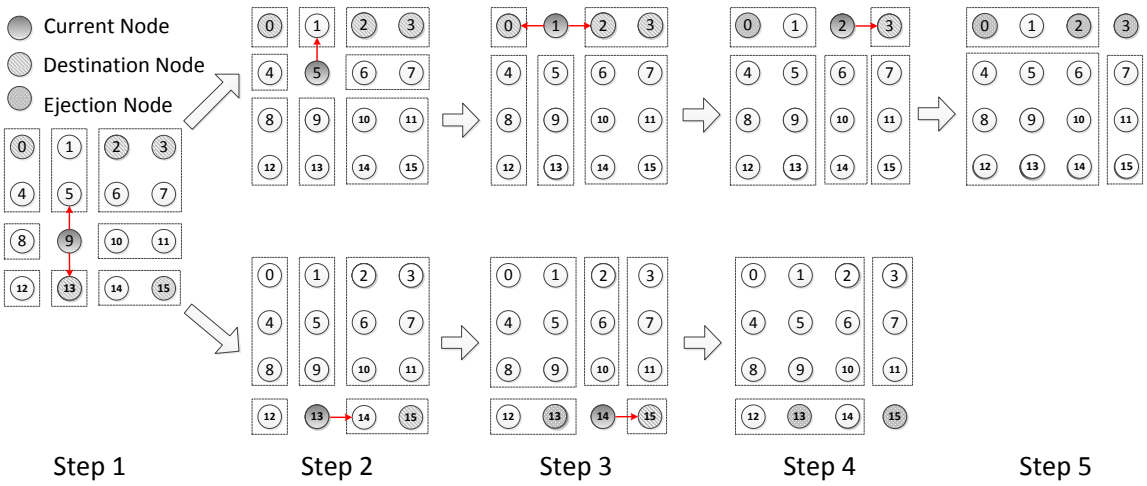


Figure 22. Multicast packet traversal

For Node 0 there will be no competition between north direction and west direction as initially there is no congestion. North direction will be preferred for Node 0. Similarly as Nodes 2 and 3 lies in partition 0, north direction will be preferred again. In the next step packet that was replicated to south, will be ejected for destination node 13 and replicated to east for destination node 15 which in the next step will reach the

destination and will be ejected. Similarly in the case of the packet replicated to north direction will further move north as load across all direction are equal. The packet then will be replicated to east and west direction for destination 0 and 2,3 respectively. In the next step packet replicated to west will be ejected from network. The east packet will be ejected from node 2 and forwarded towards further east for node 3 and will be subsequently removed from network in next step. Now let's assume a network with congestion as shown in Figure 23.

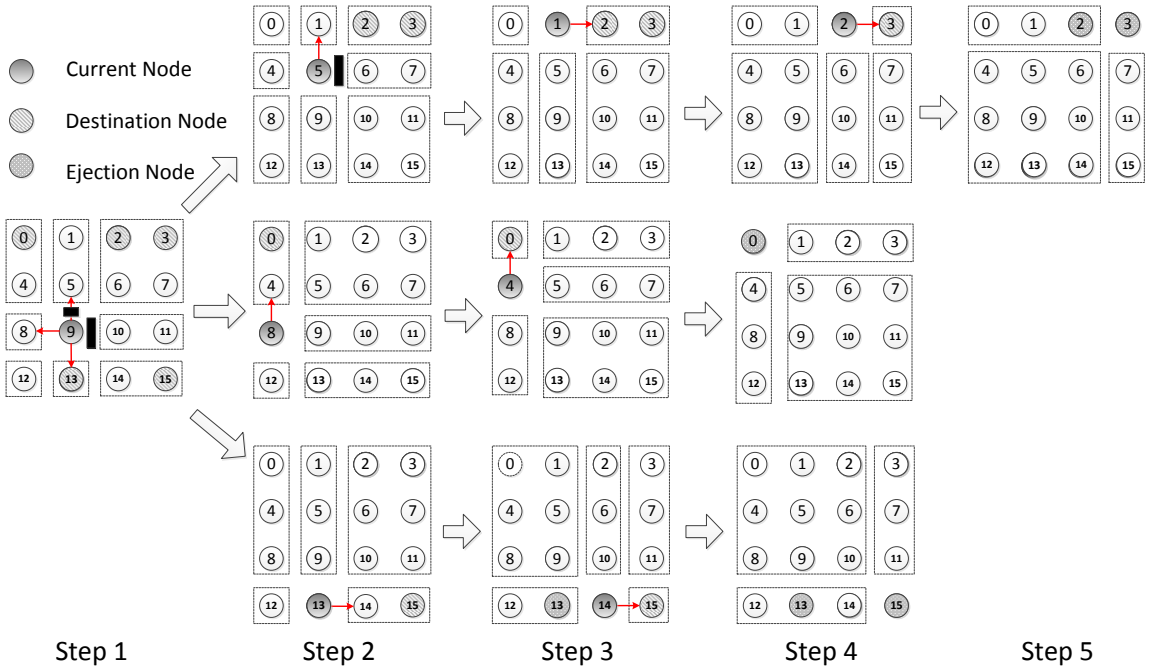


Figure 23. Multicast packet traversal in congested network. (Size of solid black box is proportional to local congestion)

Again Node 9 initiates one multicast packet with same destination list. We have the same set of partitions again. Based on the routing rules, for destinations 13 and 15,

Node 9 will replicate packet in south direction. As there is no destination in partition 7, South direction wins the right to send the packet to node 15.

For Node 0 there will be a competition between north direction and west direction, assuming west direction is less congested than north direction so west direction will carry the packet for Node 0. Similarly as Nodes 2 and 3 lies in partition 0, north will compete with East direction; assuming north direction is less congested than East, North direction will carry packet for destinations 2 and 3. In the Next step packet that was replicated to south, will be ejected for destination node 13 and replicated to east for destination node 15 which in the next step will reach the destination and will be ejected. Similarly in the case of the packet replicated to west direction, will take a path towards north for destination node 0 and will be ejected from network in the next step. For packet that went north from node 5 will move further north and will take east direction then. It will be ejected for node 2 and forwarded towards further east for node 3 and will be subsequently removed from network in next step.

D. Deadlock Avoidance

With no apparent turn restrictions on packets in B-RPM, traffic in the same network may cause cyclic dependencies which will eventual lead to stalling of the network as no packet will be able to make progress towards its destination. To avoid such situation, turn restrictions must be imposed so that network does not have any cyclic dependency. For example XY routing, is a deadlock free routing. It achieves

deadlock freedom by restricting packet traveling in Y dimension to make a turn in X dimension. Problem with such routing algorithms is that they tend to support one dimension more over another which may cause a network hotspot. If somehow we have a complemented routing for XY routing, i.e. YX routing, running on the same network than we can have a balanced routing.

By using virtual network technique single physical network can be divided in multiple virtual network; each of those network can implement deadlock free routing algorithm [22]. As shown in Figure 12, like RPM [10] we have two virtual networks for same physical network, one for the north bound traffic other for the south bound traffic. However as mentioned in BAM [13], allocation in this manner can lead to buffer imbalance between horizontal and vertical channels and also buffers are underutilized. To understand buffer underutilization let's take an example. As shown in Figure 24(a), 3 packets belonging to VN 0 are waiting router's west input port. Packet 1 and 2 will be moving north from this router, whereas Packet 3 will be forwarded to further east. The router north to current router has its south input port already occupied. With each virtual network having only 2 buffers, Packet 1 and 2 will occupy the buffers in VN 0 and packet 3 has to wait for north link of current router to get cleared so that Packet 1 and 2 can be forwarded to north. As shown in Figure 24(b), even though we don't have any traffic for VN1 and we have free buffers in VN1, Packet 3 still has to wait in upstream router. So we are essentially underutilizing the free buffers we have in hand.

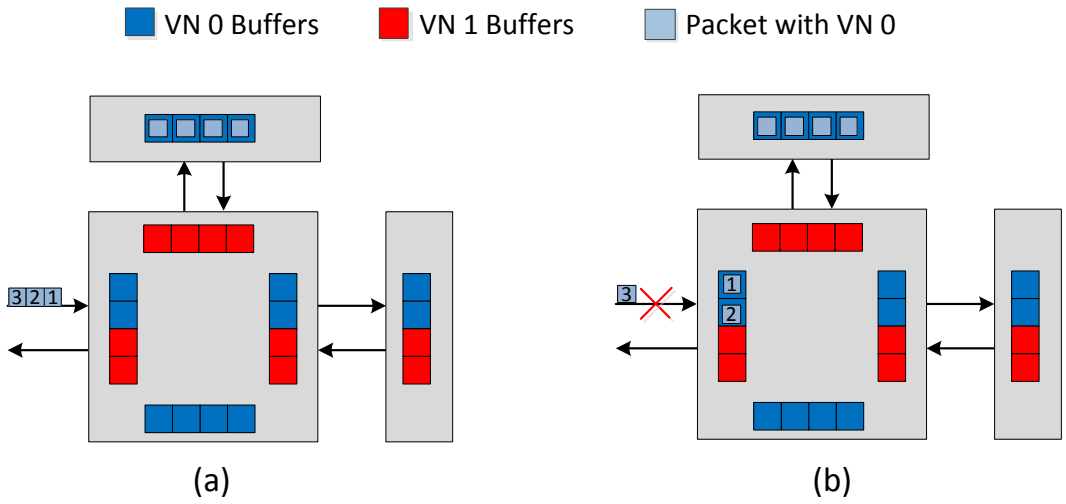


Figure 24. Virtual network example

In B-RPM, by the use of adaptive routing and relaxed virtual network allocation, explained in section "Virtual Network Allocation" section, both buffer imbalance and underutilization issues are mitigated to a good extent. However in order to minimize these issues to the maximum possible limit, we introduce "Dynamically sized Virtual Network".

1. Dynamically sized Virtual Networks

As shown in Figure 24, virtual networks are defined with the fixed set of buffers allocated to each virtual network, which often results in underutilization of buffers and buffer imbalance. For example, generally most of the time in a network one direction dominates its counter direction in terms of traffic flow in some regions. While in some other regions the other direction may be dominating like in the north side of network the

amount of traffic going further north is usually less in comparison to the traffic going south and vice versa. Assigning both virtual networks equal buffers is not a good idea. Another example may be the case of certain traffic patterns that favors one direction over others. One solution can be fixing the buffers amongst virtual networks proportionate to traffic ratio, but this will make the network rigid, and will be a useful only in case of that traffic pattern.

To counter the above mentioned issues, we propose "Dynamically sized Virtual Network" (DSVN). DSVN works by allocating virtual channels to virtual network on demand. The scheme molds itself to the current traffic trend and maximizes the buffer usage and provides effective deadlock avoidance mechanism. As virtual network's main purpose is to avoid deadlock, we need to make sure that at least we have one dedicated virtual channel per virtual network. The rest of the VCs remaining can be called as VCs in free pool. In our case we have 2 virtual networks, with 4 virtual channels, so we have 2 dedicated VCs, 1 for each virtual network and 2 in free pool. VCs from free pool can join any virtual network, if there is a demand for that virtual network. Once that VC is not in use anymore it will be returned to free pool again and will be available for any virtual network that demands it. Dedicated VC in each virtual network ensures a smooth flow of traffic in case of a potential deadlock situation amongst free pool VCs. As it implements the deadlock free routing algorithm, eventually we will have a dedicated VC available and can be used for the packet involved in the deadlock. With DSVN, effectively a VN has at least 3 Virtual channels to choose from in order to reach the destination.

In order to understand DSVN, let's take the example from last section again. Let's say we have the same scenario as shown in Figrue 24(a) in Figure 25(a). Packet 1 can move to the current router into the dedicate VN0 buffer (Figure 25(b)). In order for Packet 2 to be transferred to the current router, VN0 acquire a buffer from free pool and Packet 2 is transferred. Same goes for Packet 3; as we have one free buffer still left we can transfer the free buffer to VN0. This step is shown in Figure 25(c). Packet 3 can now travel further east as buffers in east router are available returning the occupied buffer back to free pool as shown in Figure 25(d).

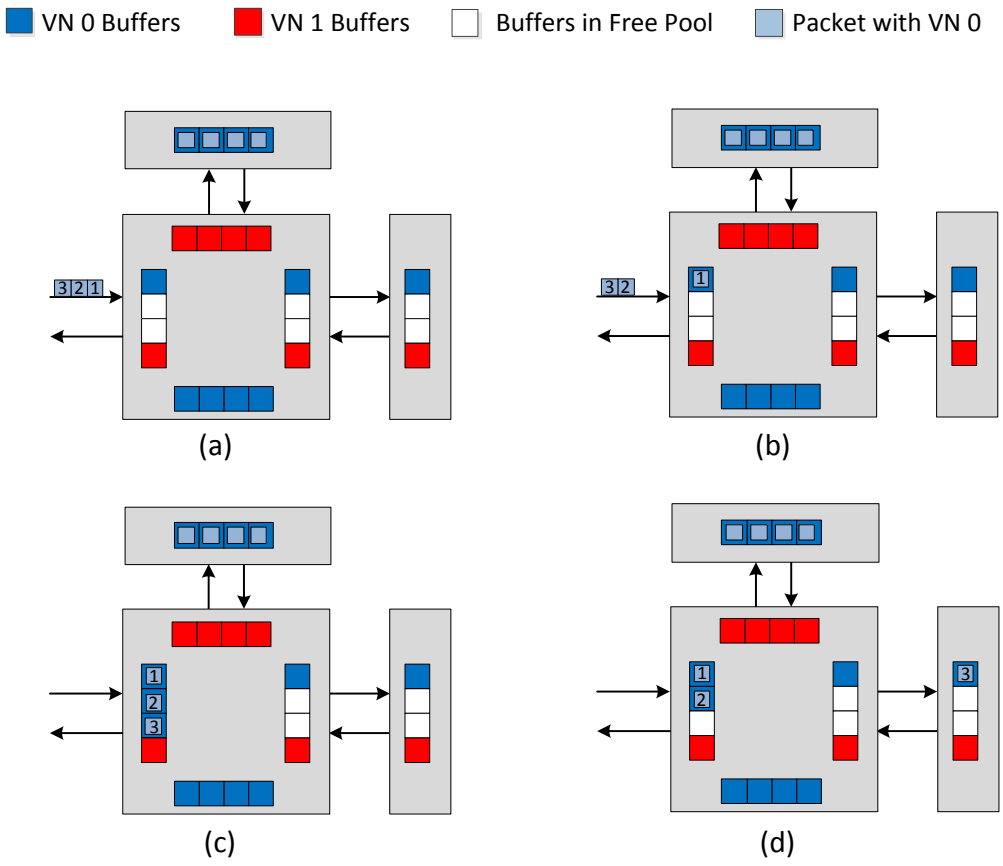


Figure 25. Dynamically sized virtual network example

2. Virtual Network Allocation

As a single physical network is divided into two virtual networks, packets have to be allocated the virtual network. Rules for assignment of Virtual Network are a bit relaxed in comparison with RPM [10]. As with RPM [10] north bound traffic will be assigned Virtual Network 0 and south bound traffic will be assigned Virtual Network 1. However in case of replicated packets with all destinations exactly horizontal to the current router i.e. in partition 3 or partition 7, packets are allowed to switch their VN depending on the congestion state of each VNs. For example a replicated packet with its destinations in partition 7 only, has inherited VN1 from its source packet, but the current router realizes that VN1 is congested so it can move the packet from VN1 to VN0. This rule does not apply for a replicated packet with destinations list having nodes in partition 7 and partition 0 or 6.

CHAPTER V

EXPERIMENTAL RESULTS

This chapter compares the performance of B-RPM with its predecessors. The chapter evaluates and compares the algorithm on various accounts with other multicast algorithm but before getting into detailed performance analysis let's start with simulation configuration.

A. Experimental Methodology

To evaluate B-RPM, we use a cycle accurate interconnect simulator. The simulator models all router delays and wire latencies precisely. For the purpose of simulation, interconnect simulator is configured with 2D mesh topology with varying number of nodes (36, 64, 100, 144, 196 and 256). Router used is the state-of-the-art 2 stage router with 4 VCs per physical channel and 4-flit as VC depth. Link bandwidth is defined as 128 bits. Performance evaluation is done using synthetic workloads like Uniform random (UR), Transpose (TP) and Bit Complement (BC). However for multicast packets, destinations in destination list are uniformly distributed rather than defined by traffic pattern. In order to estimate dynamic and static power consumption we use Orion [50]. Orion is configured with 4GHZ clock frequency for router and link, power supply as 1V, 65nm technology and 50% switch activity. Table III summarizes the configuration for simulator.

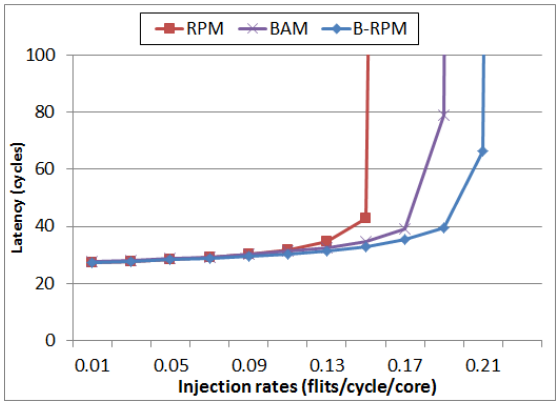
Table III. Simulator configuration

Feature	Baseline Configuration	Other Configurations
Topology	8 x 8 Mesh	6 x 6, 10 x 10, 12 x 12, 14 x 14, 16 x 16 Mesh
Routing Algorithms	RPM, BAM, B-RPM	-
Router Delay	2 Cycles	-
Virtual Channels per Port	4	-
Virtual Channel Depth	4	-
Packet Length	4 flits	-
Workload Type	Uniform Random	Transpose, Bit Complement
Ratio of Multicast Traffic	10%	5%, 20%, 40%, 80%
Average Destination List Size	16 (uniformly distributed)	4, 8, 32
Simulation Warmup Cycles	10,000	-
Total Simulation Cycles	20,000	-

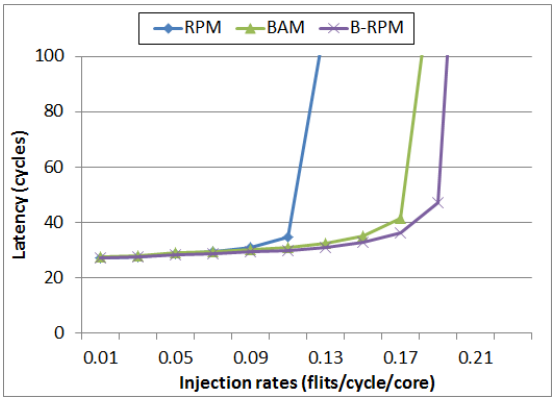
B. Performance Evaluation

Performance of on-chip network is evaluated on the basis of average communication latency. Figure 26 shows the results on an 8×8 network with three synthetic traffic patterns. The trend in three traffic patterns is the same. With high injection rates, B-RPM reduces communication latency by 20% on average compared with BAM. For the network saturation, B-RPM postpone saturation to higher loads, 10% higher than BAM and 30% higher than RPM, which means B-RPM can deliver high network throughput. Compared with RPM, B-RPM considers the local congestion information and explores the path diversity, which balances the workloads between horizontal and vertical links. Although BAM also hires adaptive routing, at high loads it suffers from congestion in escape channels. Since in multicast traffic a packet targets for

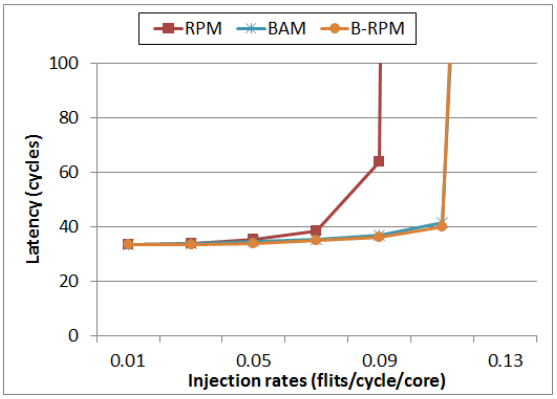
multiple output ports of the same router, a packet will hold the router resources until all the replicas of the packet are sent to the requested output ports. When the injection rate becomes high, more packets are forced into escape channels. Because only a small amount of VCs are used as escape channels, escape VCs become the bottleneck of the network and hurts the whole network performance. To overcome this issue B-RPM hires DSVN that efficiently utilize the resources.



(a) Uniform Random



(b) Transpose



(c) Bit Complement

Figure 26. Network performance with 3 synthetic traffic pattern. (8 x 8 mesh network, 10% multicast traffic, maximum 16 destination)

We also explore how the number of escape channels affects the network performance, as shown in Figure 27. It is observed that as the number of escape channels increases, the network throughput of BAM improves, but still worse than B-RPM with only 4 VCs. We can see that BAM with 4 normal VCs and 2 escape channels has the similar saturation point as B-RPM but the communication latency of B-RPM at high injection rate is still 20% less than BAM. However, increasing the number of escape channels will introduce hardware overhead.

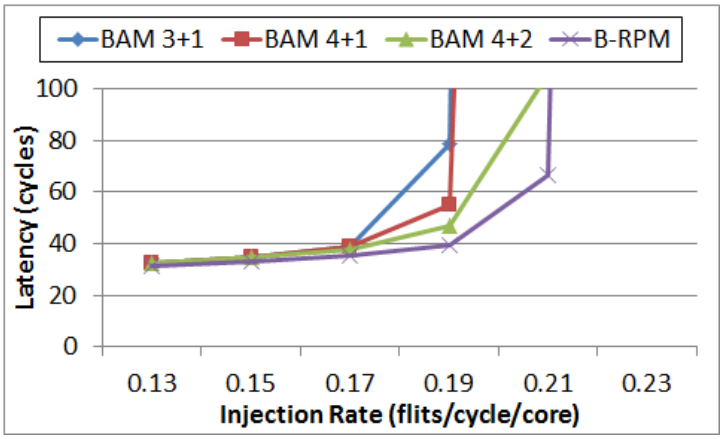


Figure 27. Packet latency comparison of B-RPM with different configuration of BAM. (8 x 8 mesh network, 10% multicast traffic, maximum 16 destination)

Then we analyze the impact of DSVN on both RPM and B-RPM in Figure 28. Original RPM uses the traditional virtual networks to handle deadlock. When replaced with DSVN, the network throughput of RPM improves dramatically. The reason is that original RPM suffers from resource underutilization and imbalanced horizontal and vertical workloads. DSVN relieves those two problems by balancing the resource allocation between two VNs. On average the network throughput of RPM with DSVN

increases by 12%. While RPM enjoys a high network throughput with DSVN, B-RPM with traditional virtual networks has a similar saturation point compared to B-RPM. That is because, compared with RPM, B-RPM explores the path diversity of the network and achieves a more balanced network. On the other hand, DSVN helps B-RPM reduce communication latency by 15% at high workloads since DSVN can dynamically change the resource allocation in virtual networks, therefore make the resource utilization more efficient.

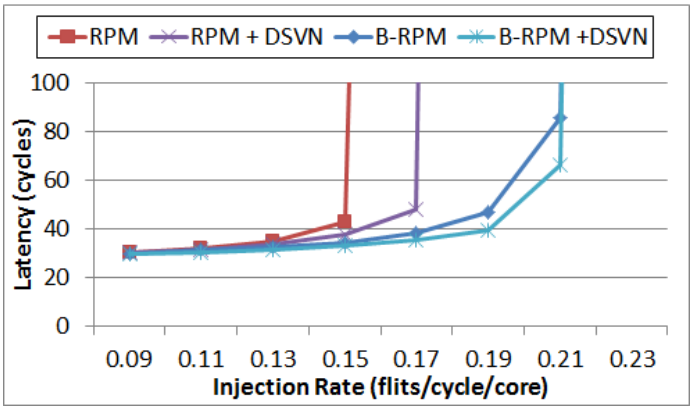


Figure 28. Packet latency comparison of B-RPM and RPM with and without DSVN. (8 x 8 mesh network, 10% multicast traffic, maximum 16 destination)

C. Header Size Analysis

Figure 29 presents the comparison amongst commonly used header format and proposed unicast differentiated header format. It's pretty much obvious that bit vector is the most expensive header format. For smaller networks more or less all other schemes have the same header size but as the network size grows, for low average destinations,

destination ID list and unicast differentiated compressed scheme appear as winners. Unicast differentiated compressed average header size is 60% less in comparison to that of compressed header. However it all boils down to the design time complexity designer is willing to afford. Pure destination ID list scheme needs a programmed logic as it needs to read the count first and then parse the list of destination accordingly. Such schemes are almost impossible to implement within 1 cycle limit. Compressed header offers some complexity in terms of design as each router will have a different header parsing logic. Most simple scheme, after bit vector with almost the same header size, as more complex schemes, is Unicast differentiated bit vector. For low average destinations and small ratio of multicast packet, the results from this scheme are almost the same as other schemes and that too with a very less complex design.

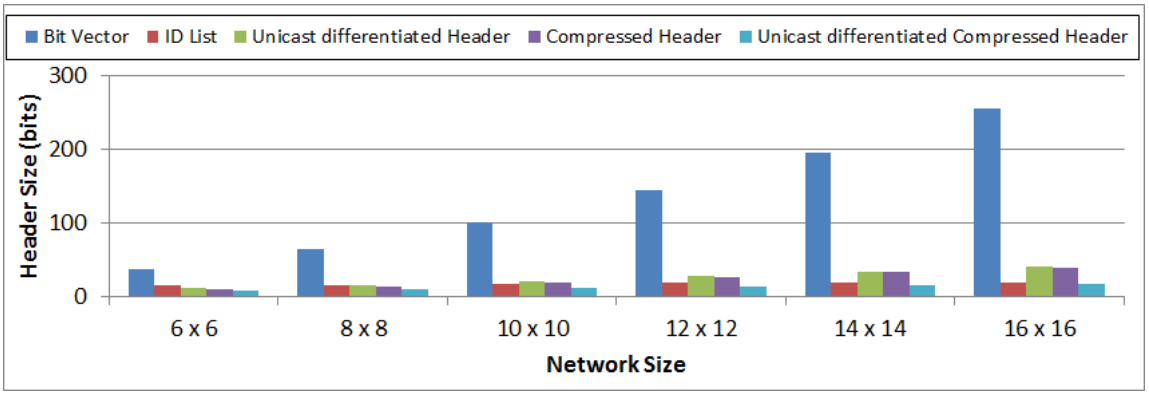
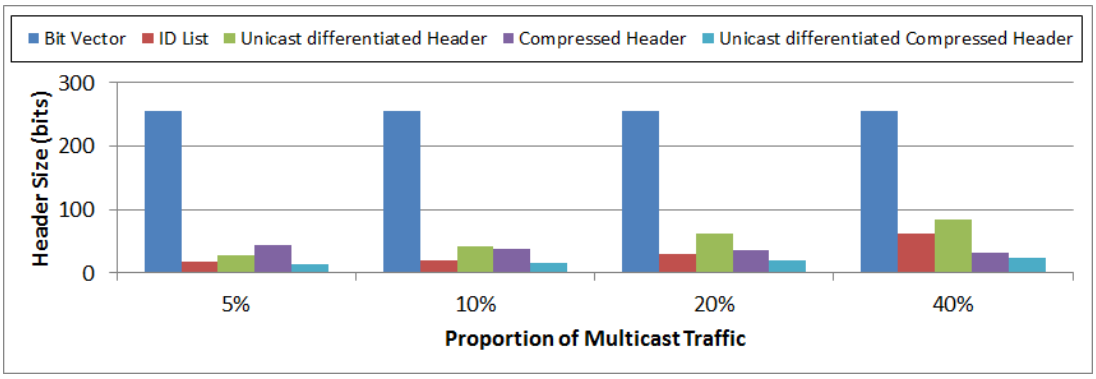


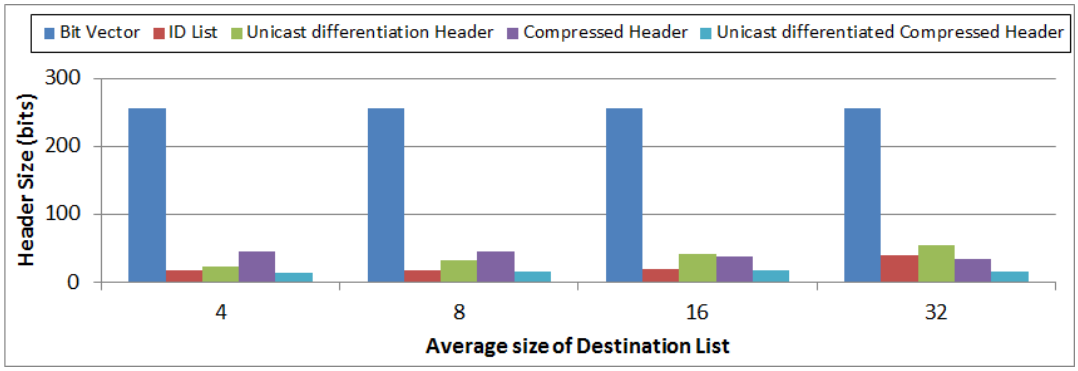
Figure 29. Header size comparison with varying network size. (10% multicast traffic, maximum 16 destination)

However, as the ratio of multicast traffic grows the situation changes. With more multicast packets, more packets will have complete bit vectors in their headers and the average header size will grow in case of unicast differentiated bit vector. Same is the

case with growing size of destination list. With large size of destination list, replication of multicast packet will result in more multicast packets and again will affect the average header size. As the number of average destinations grows or multicast to unicast ratio increases one may prefer schemes like compressed header or unicast differentiated compressed header scheme. Figure 30(a) and 30(b) presents comparison of schemes with large ratio of multicast to unicast traffic and increased size of average destination list.



(a) Header size comparison with varying proportion of multicast traffic. (16 x 16 mesh network, maximum 16 destinations)



(b) Header size comparison with varying size of destination list. (16 x 16 mesh network, 10% multicast traffic)

Figure 30. Header size comparison with varying destination list size and ratio of multicast traffic

D. Power Analysis

B-RPM algorithm doesn't enjoy a significant power benefit over its predecessors. At low loads the power benefit is less than 2%. At higher loads since most of the algorithms saturate the network and the link activity is almost at halt and B-RPM is still at work, it consumes more power than the other algorithm. Figure 31 shows the power comparison amongst the three algorithms.

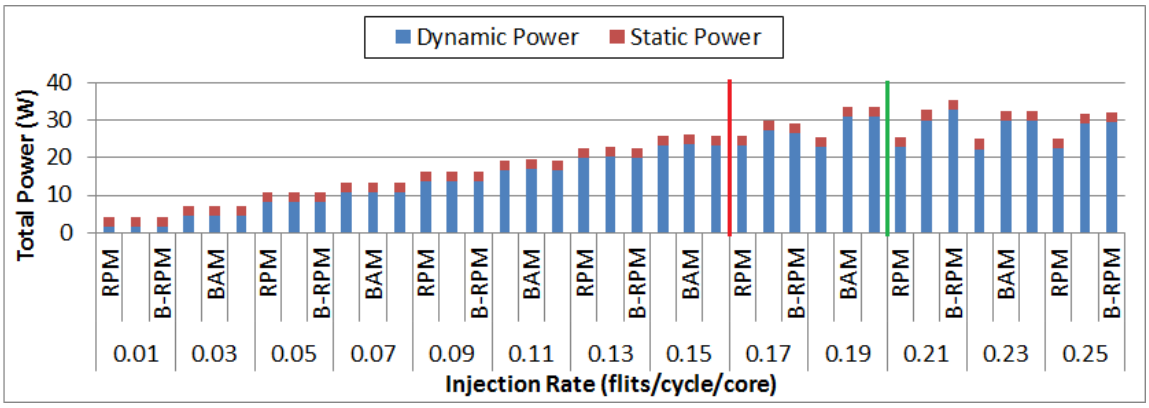


Figure 31. Power consumption analysis. (8x8 mesh with maximum 16 destinations and 10% multicast traffic) (Red line indicates saturation point of RPM and green line indicates saturation point of BAM)

Figure 32 highlights the dynamic power consumption by different components of OCN. Power consumed by Link has a dominating share in total dynamic power consumption.

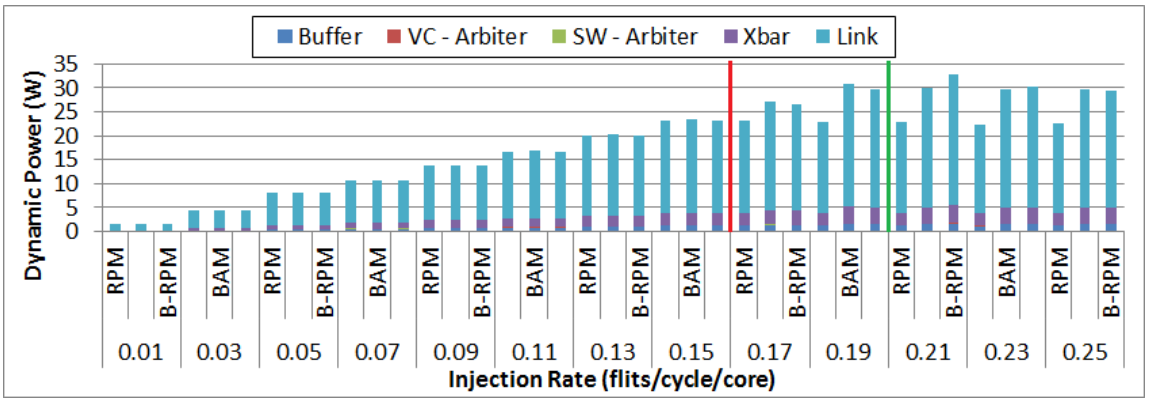


Figure 32. Power consumption by different components of OCN. (8x8 mesh with maximum 16 destinations and 10% multicast traffic) (Red line indicates saturation point of RPM and green line indicates saturation point of BAM)

One can see from Figure 32 that once the algorithm reaches saturation, link activity saturates as well. This can also be established by Figure 33 which shows the average link utilization of the algorithms.

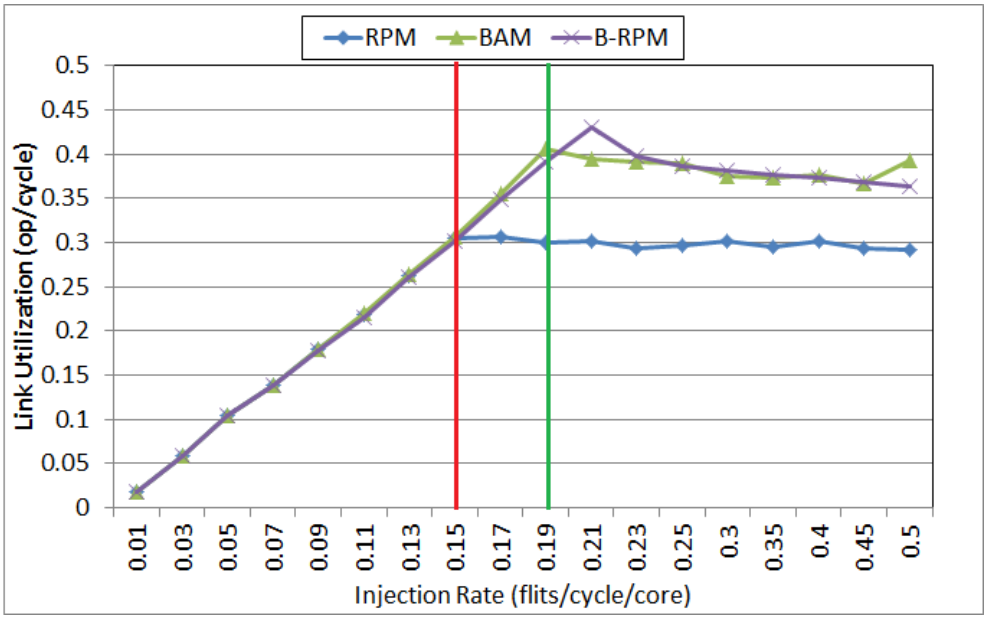


Figure 33. Average link utilization. (8x8 mesh with maximum 16 destinations and 10% multicast traffic) (Red line indicates saturation point of RPM and green line indicates saturation point of BAM)

Though the algorithm itself is not able to pull any power advantage but reduced header size for sure can lessen the link activity and save some power for us. For smaller networks the unicast differentiated compressed header consumes 15% less link power than consumed by standard bit vector. For larger network it saves up to 24% of dynamic link power. In comparison to compressed header, unicast differentiated compressed header saves 5% of dynamic link power. Figure 34 shows the average dynamic link power consumed by head flit in different schemes.

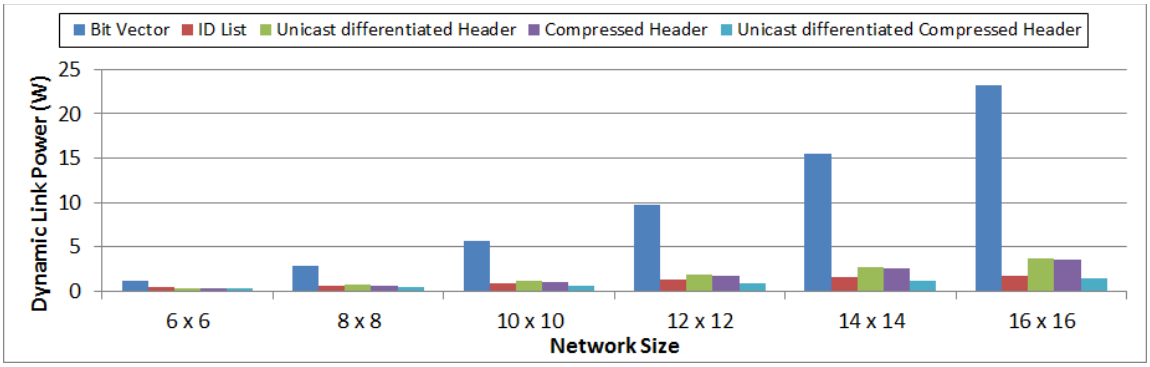


Figure 34. Dynamic link power consumption by head flit. (maximum 16 destinations and 10% multicast traffic)

E. Scalability

To measure the scalability we evaluated the algorithms on 3 criteria: network size, proportion of multicast traffic and average size of destination list. Figure 35 summarizes the network performance of algorithms with varying network size. In small networks, performance of all algorithms is almost the same. However, in large networks this is not the case. In large networks, more nodes are generating packets at the same

time, and also average hop count is high. Therefore, packets spend more time in the network. In this situation, balancing the workload among links becomes more important. Compared with B-RPM, the performance of RPM and BAM degrades sharply when the network size increases.

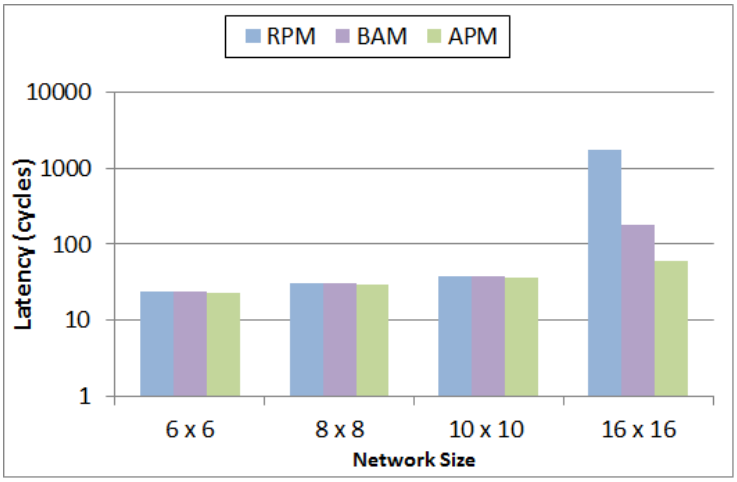


Figure 35. Scalability to network size. (10% of multicast traffic, maximum 16 destinations)

Almost similar situation can be found when we increase the ratio of multicast to unicast packet. When the ratio is less than 20% all algorithms perform the same. But as soon as the ratio leaves this range, RPM and BAM degrades sharply, while B-RPM performance stays stable till 40% of multicast packets. That is because B-RPM balances the workload better than RPM, and utilizes the buffer resources more efficient than BAM. Figure 36 summarizes the performance with varying ratio of multicast packets.

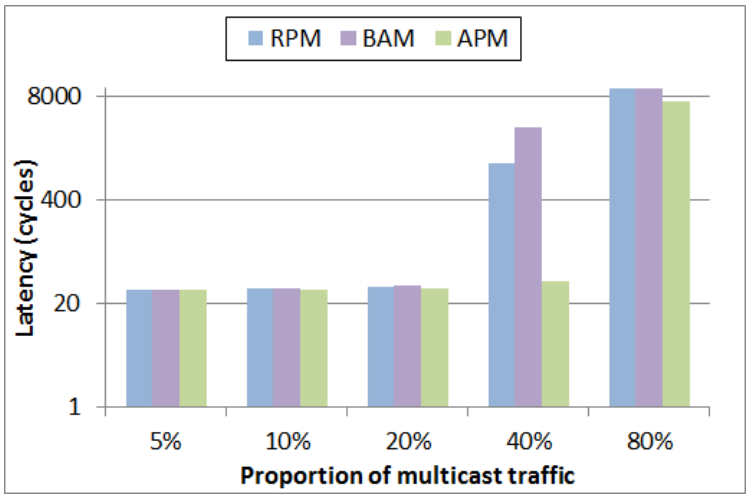


Figure 36. Scalability to proportion of multicast traffic. (8 x 8 Mesh network, maximum 16 destinations)

For large number of destination per multicast packet, again B-RPM proves itself to be better. Since multicast packets replicate during the course of their traversal, they generate more multicast packets with larger destination list. Larger the number of destinations, more multicast packets are generated. So the actual ratio of multicast to unicast packets increases as the number of destinations increases. Figure 37 shows the comparison on how the algorithms react to growing number of destinations per multicast packets. B-RPM is again more scalable in handling the number destinations per multicast packet than other algorithms.

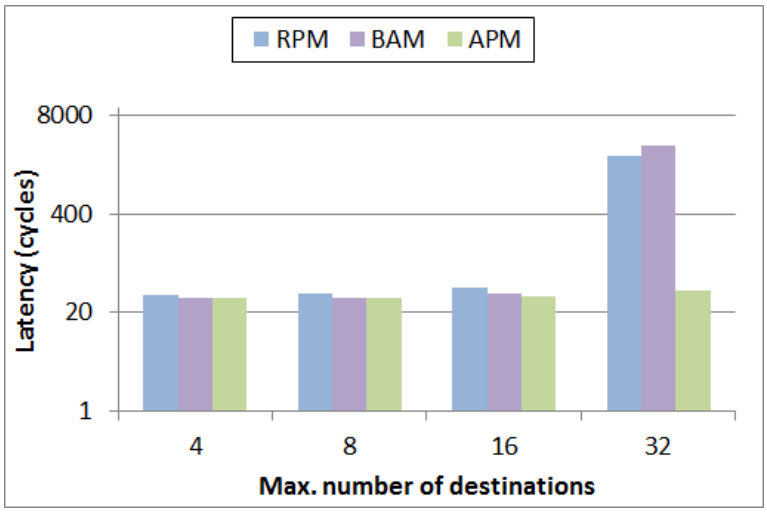


Figure 37. Scalability to maximum number of destinations. (8 x 8 mesh network, 10% multicast traffic)

CHAPTER VI

CONCLUSIONS

With on-chip networks replacing buses in CMPs as default communication architecture, it should be ascertained that on-chip networks can handle all kind of generated traffic. One common type of message in case of CMPs is multicast message. Multicast packet generates lot of packets while traversing network thus increasing the load of network exponentially. The routing algorithm must handle such situation.

In this paper, we propose an adaptive multicast routing, B-RPM, to improve network performance in a mix of one-to-one and one-to-many traffic and balance the workloads among the links. In B-RPM, we use DSVN to provide deadlock avoidance. Besides providing deadlock avoidance, DSVN allocates network resources into different virtual networks according to the run-time network status, delivering better resources utilization than the traditional virtual network scheme. We also explore various options for representing destination list in head flit. Findings indicate that only by differentiating the unicast and multicast packets we can reduce the average header size to a great extent. For results, the proposed algorithm improves network latency at high load up to 50% and saturation throughput up to 18% over the most recent multicast algorithm. Also with new header representation scheme dynamic link power is saved up to 24 %.

In future we plan to evaluate our scheme in full system simulator with real world traffic patterns and analyze the overall performance of system.

REFERENCES

[1] "The International Technology Roadmap for Semiconductors," http://www.itrs.net/.

[2] Y. Hoskote, S. Vangal, A. Singh *et al.*, "A 5-GHz Mesh Interconnect for a Teraflops Processor," *IEEE Micro,* vol. 27, no. 5, pp. 51-61, 2007.

[3] D. Wentzlaff, P. Griffin, H. Hoffmann *et al.*, "On-Chip Interconnection Architecture of the Tile Processor," *IEEE Micro,* vol. 27, no. 5, pp. 15-31, 2007.

[4] K. Sankaralingam, R. Nagarajan, L. Haiming *et al.*, "Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture," in *Proc. 30th Annual International Symposium on Computer Architecture*, 2003, pp. 422-433.

[5] P. Gratz, S. Karthikeyan, H. Hanson *et al.*, "Implementation and evaluation of a dynamically routed processor operand network," in *Proc. 1st International Symposium on Networks-on-Chip*, 2007, pp. 7-17.

[6] M. Bedford Taylor, W. Lee, S. Amarasinghe *et al.*, "Scalar operand networks: on-chip interconnect for ILP in partitioned architectures," in *Proc. 9th International Symposium on High-Performance Computer Architecture*, 2003, pp. 341-353.

[7] J. L. Hennessy, and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Waltham, MA: Morgan Kaufmann Pub, 2011.

[8] M. M. K. Martin, M. D. Hill, and D. A. Wood, "Token Coherence: decoupling performance and correctness," in *Proc. 30th Annual International Symposium on Computer Architecture*, 2003, pp. 182-193.

[9] N. E. Jerger, P. Li-Shiuan, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *Proc. 35th International Symposium on Computer Architecture*, 2008, pp. 229-240.

[10] W. Lei, J. Yuho, K. Hyungjun *et al.*, "Recursive partitioning multicast: A bandwidth-efficient routing for Networks-on-Chip," in *Proc. 3rd ACM/IEEE International Symposium on Networks-on-Chip*, 2009, pp. 64-73.

[11] N. D. E. Jerger, L.-S. Peh, and M. H. Lipasti, "Virtual tree coherence: Leveraging regions and in-network multicast trees for scalable cache coherence,"

in *Proc. 41st Annual IEEE/ACM International Symposium on Microarchitecture*, 2008, pp. 35-46.

[12] S. Rodrigo, J. Flich, J. Duato *et al.*, "Efficient unicast and multicast support for CMPs," in *Proc. 41st Annual IEEE/ACM International Symposium on Microarchitecture*, 2008, pp. 364-375.

[13] S. Ma, N. E. Jerger, and Z. Wang, "Supporting efficient collective communication in NoCs," in *Proc. 18th IEEE International Symposium on High Performance Computer Architecture*, 2012, pp. 1-12.

[14] P. Abad, V. Puente, and J. A. Gregorio, "MRR: Enabling fully adaptive multicast routing for CMP interconnection networks," in *Proc. 15th IEEE International Symposium on High Performance Computer Architecture*, 2009, pp. 355-366.

[15] T. Krishna, L.-S. Peh, B. M. Beckmann *et al.*, "Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication," in *Proc. 44th Annual IEEE/ACM International Symposium on Microarchitecture*, 2011, pp. 71-82.

[16] W. Lei, P. Kumar, R. Boyapati *et al.*, "Efficient lookahead routing and header compression for multicasting in networks-on-chip," in *Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2010, pp. 1-10.

[17] W. J. Dally, and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. Design Automation Conference*, 2001, pp. 684-689.

[18] W. J. Dally, and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.

[19] L. S. Peh, and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. 7th International Symposium on High-Performance Computer Architecture*, 2001, pp. 255-266.

[20] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*. San Francisco, CA: Morgan Kaufmann, 2003.

[21] W. J. Dally, and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers,* vol. 100, no. 5, pp. 547-553, 1987.

[22] M. Chaudhuri, and M. Heinrich, "Exploring virtual network selection algorithms in DSM cache coherence protocols," *IEEE Transactions on Parallel and Distributed Systems,* vol. 15, no. 8, pp. 699-712, 2004.

[23] K. V. Anjan, and T. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: DISHA," in *Proc. 22nd Annual International Symposium on Computer Architecture*, 1995, pp. 201-210.

[24] D. R. Kumar, W. A. Najjar, and P. K. Srimani, "A new adaptive hardware tree-based multicast routing in k-ary n-cubes," *IEEE Transactions on Computers,* vol. 50, no. 7, pp. 647-659, 2001.

[25] L. Xiaola, P. K. McKinley, and L. M. Ni, "Deadlock-free multicast wormhole routing in 2-D mesh multicomputers," *IEEE Transactions on Parallel and Distributed Systems,* vol. 5, no. 8, pp. 793-804, 1994.

[26] M. P. Malumbres, J. Duato, and J. Torrellas, "An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors," in *Proc. 8th IEEE Symposium on Parallel and Distributed Processing*, 1996, pp. 186-189.

[27] R. Sivaram, D. K. Panda, and C. B. Stunkel, "Efficient broadcast and multicast on multistage interconnection networks using multiport encoding," *IEEE Transactions on Parallel and Distributed Systems,* vol. 9, no. 10, pp. 1004-1028, 1998.

[28] C. B. Stunkel, J. Herring, B. Abali *et al.*, "A new switch chip for IBM RS/6000 SP systems," in *Proc. ACM/IEEE Conference on Supercomputing*, 1999, pp. 16-16.

[29] J. S. Turner, "An optimal nonblocking multicast virtual circuit switch," in Proc. 13th IEEE INFOCOM, Networking for Global Communications, Year, pp. 298-305 vol.1.

[30] "Intel Nehalem," http://www.realworldtech.com/page.cfm?ArticleID=RWT082807020032.

[31] P. Conway, N. Kalyanasundharam, G. Donley *et al.*, "Cache Hierarchy and Memory Subsystem of the AMD Opteron Processor," *IEEE Micro,* vol. 30, no. 2, pp. 16-29, 2010.

[32] K. Strauss, X. Shen, and J. Torrellas, "Uncorq: Unconstrained snoop request delivery in embedded-ring multiprocessors," in *Proc. 40th Annual IEEE/ACM International Symposium on Microarchitecture*, 2007, pp. 327-342.

[33] J. Laudon, and D. Lenoski, "The SGI origin: a ccNUMA highly scalable server," in *Proc. 24th Annual International Symposium on Computer Architecture*, 1997, pp. 241-251.

[34] E. E. Bilir, R. M. Dickson, H. Ying *et al.*, "Multicast snooping: a new coherence method using a multicast address network," in *Proc. 26th International Symposium on Computer Architecture*, 1999, pp. 294-304.

[35] M. M. K. Martin, P. J. Harper, D. J. Sorin *et al.*, "Using destination-set prediction to improve the latency/bandwidth tradeoff in shared-memory multiprocessors," *SIGARCH Comput. Archit. News,* vol. 31, no. 2, pp. 206-217, 2003.

[36] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proc. 31st Annual International Symposium on Computer Architecture*, 2004, pp. 188.

[37] P. Abad, V. Puente, Jos *et al.*, "Rotary router: an efficient architecture for CMP interconnection networks," in *Proc. 34th Annual International Symposium on Computer Architecture*, 2007, pp. 116-125.

[38] A. Kumary, P. Kunduz, A. P. Singhx *et al.*, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *Proc. 25th International Conference on Computer Design*, 2007, pp. 63-70.

[39] L. Zhonghai, Y. Bei, and A. Jantsch, "Connection-oriented multicasting in wormhole-switched networks on chip," in *Proc.* IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures, Year, pp. 205-2011.

[40] Y. H. Kang, J. Sondeen, and J. Draper, "Multicast routing with dynamic packet fragmentation," in *Proc. 19th ACM Great Lakes symposium on VLSI*, 2009, pp. 113-116.

[41] F. Samman, T. Hollstein, and M. Glesner, "New Theory for Deadlock-Free Multicast Routing in Wormhole-Switched Virtual-Channelless Networks-on-Chip," *IEEE Transactions on Parallel and Distributed Systems*, no. 99, pp. 1-1, 2011.

[42] X. Wang, M. Yang, Y. Jiang *et al.*, "On an efficient NoC multicasting scheme in support of multiple applications running on irregular sub-networks," *Microprocessors and Microsystems,* vol. 35, no. 2, pp. 119-129, 2011.

[43] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems,* vol. 4, no. 12, pp. 1320-1331, 1993.

[44] D. E. Culler, A. Gupta, and J. P. Singh, *Parallel Computer Architecture: A Hardware/Software Approach*, 1st ed. San Francisco, CA: Morgan Kaufmann Publishers Inc., 1997.

[45] C. Chiang, and L. Ni, "Deadlock-free multi-head wormhole routing," in *Proc. 1st High Performance Computing-Asia*, 1995.

[46] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proc. 14th IEEE International Symposium on High Performance Computer Architecture*, 2008, pp. 203-214.

[47] K. Jongman, P. Dongkook, T. Theocharides *et al.*, "A low latency router supporting adaptivity for on-chip interconnects," in *Proc. 42nd Design Automation Conference*, 2005, pp. 559-564.

[48] S. Ma, N. E. Jerger, and Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *Proc. 38th Annual International Symposium on Computer Architecture*, 2011, pp. 413-424.

[49] C. Chiang, and L. Ni, "Multi-address encoding for multicast," *Parallel Computer Routing and Communication*, pp. 146-160, 1994.

[50] H.-S. Wang, X. Zhu, L.-S. Peh *et al.*, "Orion: a power-performance simulator for interconnection networks," in *Proc. ACM/IEEE International Symposium on Microarchitecture*, 2002, pp. 294-305.

VITA

Name:	Noman Shaukat
Address:	15325 Redmond Way Apt # B205, Redmond, WA 98052
Email Address:	qnomans2k@tamu.edu
Education:	B.E., Computer & Information Systems, NEDUET, 2007
	M.S., Computer Engineering, Texas A&M University, 2012