

Multicast Routing with Dynamic Packet Fragmentation

Young Hoon Kang
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292 USA
youngkan@ISI.EDU

Jeff Sondeen
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292 USA
jsondeen@ISI.EDU

Jeff Draper
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292 USA
draper@ISI.EDU

ABSTRACT

Networks-on-Chip (NoCs) become a critical design factor as chip multiprocessors (CMPs) and systems on a chip (SoCs) scale up with technology. With fundamental benefits of high bandwidth and scalability in on-chip networks, a newly added multicast capability can further enhance the performance by reducing the network load and facilitate coherence protocols of many-core CMPs [10]. This paper proposes a novel multicast router with dynamic packet fragmentation in on-chip networks. Packet fragmentation is performed to avoid deadlock in blocking situations, releasing the hold of an output virtual channel (VC) and allowing another packet to use the freed VC. From circuit simulation of the design implemented with IBM 90nm technology, the proposed router reduces latency by 38.6% and consumes 9% less energy than a unicast baseline router at the baseline saturation.

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Packet-switching networks

General Terms: Performance, Design

Keywords: NoC, Interconnection Network, On-chip router

1. INTRODUCTION

The increased capability of VLSI technology makes it possible to build many-core architectures such as chip multiprocessors (CMPs) on a single die. In such chips, prior research has shown that a well structured network-on-chip (NoC) can provide more flexible and scalable communication with fundamental benefits of increased bandwidth and reduced latency as compared to conventional buses or point-to-point links. However, there are still critical challenges in terms of power and latency [8]. Prior work [4], [8], [10] proposed that hardware multicast support could enhance performance further and enable the development of promising protocols for large numbers of cores. A single packet with multiple destinations avoids unnecessary packet replications traversing the network. The reduced network load yields a decreased average packet latency and power savings. Thus, hardware support for multicast routing is essential for on-chip networks to overcome latency and power consumption limitations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GLSVLSI'09, May 10–12, 2009, Boston, Massachusetts, USA.
Copyright 2009 ACM 978-1-60558-522-2/09/05...\$5.00.

in architectures where the frequency of multicast operations is significant.

In this paper, we present a novel multicast router for networks-on-chip. Deadlock-free multicast routing is achieved by fragmenting packets in particular blocking situations. Normally, deadlock avoidance in multicast routing is accomplished through the provision of extra virtual channels (VCs), but from the strict constraints of area and power, increased VCs are not necessarily an ideal solution [1], [11]. The proposed dynamic packet fragmentation offers deadlock avoidance without additional VCs and increases VC utilization. Since fragmentation prevents a packet from holding a channel when it has no flits available for traversal due to upstream blocking conditions, not only are deadlock conditions resolved but there is also potential for increased channel utilization. Once a packet is fragmented, the last traversing flit of the fragmented packet releases the hold of a VC thereby allowing other packets to use the freed output VC. Thus, fragmentation prevents upstream blocking from propagating to downstream routers. Congested upstream routers do not force inputs of downstream routers to throttle to a reduced rate; instead the input of a downstream router can be utilized by another packet.

The proposed router has been implemented with synthesizable VHDL code, and a physical layout was generated. From circuit simulation with synthetic workloads, the performance and average energy consumption were measured and analyzed, compared to a baseline unicast router. The results show that the proposed multicast router is superior to the baseline design, reducing latency by 38.6% and energy consumption by 9% and also providing 30% more throughput.

This paper is organized as follows. Section 2 describes various multicast routing schemes. Section 3 discusses the router architecture and operation of the proposed multicast routing scheme with dynamic packet fragmentation. In section 4, performance and energy analyses are presented, and section 5 summarizes the paper.

2. MULTICAST ROUTING SCHEMES

This section reviews prior hardware-based multicast routing schemes in off-chip and on-chip routers. Several recent proposals have suggested circuit-switched multicast routing schemes for NoCs. In Virtual Circuit Tree Multicasting (VCTM) [10], multicast data flits follow paths set up by multiple unicast+setup packets. VCTM outperforms a unicast packet-switched baseline in various environments, but multicast routing is achieved with the cost of a CAM-based virtual circuit tree. Lu, et al., suggested connection-oriented multicasting in wormhole switched networks [4]. This scheme exhibits better throughput than unicast routing, but the multicast group establishment is a significant overhead.

In packet-switched multicast routing, two major techniques are suggested: path-based and tree-based. In path-based routing, the source node partitions the destinations into several ordered lists. The multiple packet worms then visit destinations in a predefined order through disjoint paths. Since the multicast packet visits all intermediate nodes until it reaches the last destination, path-based routing typically results in high latency. In tree-based routing, the multidestination packet is routed along a common path, and the packet is copied into different channels at branches. However, tree-based multicast routing is susceptible to packets blocking at branch nodes, where such blocking increases the probability of deadlock in wormhole routed networks.

Prior literature suggested viable solutions to avoid deadlock by partitioning and systematically allocating virtual channels [1]. Kumar proposed a hardware tree-based multicast routing algorithm (HTA) with deadlock detection and a recovery mechanism [6]. When a potential deadlock is detected, the packet is routed to the deadlock queue and reinjected into the network after a predetermined amount of time. Tree-based multicast with branch pruning [3] solved deadlocks by controlling multicast through a pruning mechanism. An auxiliary buffer associated in each input channel holds data flits until the tail flit exits a node. The auxiliary buffer makes it possible for multidestination worms to branch since it prunes off the tree when any branch is blocked. The tree-based multicast with branch pruning performs well, but the auxiliary buffer size is a limitation.

3. Router Architecture and Operation

This paper proposes an efficient tree-based multicast routing scheme, which can be implemented with minimum hardware. Deadlock is avoided through a dynamic packet fragmentation technique. The following sections present more details of the architecture and operation of the proposed router.

3.1 Router Architecture and Operation

Figure 1 shows the pipeline stages of the proposed router. Both unicast and multicast packets follow the same pipeline stages of buffer writing/routing computation (BW/RC), switch allocation/virtual channel allocation (SA/VA), switch traversal (ST), and line traversal (LT) [9]. While a head flit is written to an input buffer, the routing computation starts based on the destination nodes field. To reduce the overhead of multidestination packets, two types of multidestination address encodings are adopted. The proposed idea uses bit-string encoding [7] (a simple bit vector, where each bit corresponds to a destination of the bit position.) for a reachable network arbitrarily limited to 16 nodes in a two-dimensional mesh network. (Larger numbers could be accommodated depending on how much header overhead is tolerable for a specific architecture.) Given a routing function which returns a number of output ports, the router discriminates between a unicast and multicast packet.

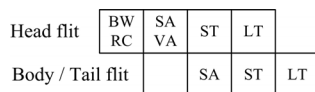


Figure 1. Router pipeline

For unicast packet routing, the result of the routing computation is used for SA and VA at the next cycle. VA is accomplished simply by finding a free output VC from the requested output port and assigning it to a SA winner as described in [5]. If SA succeeds but

VA fails, the head flit retries from SA until both SA and VA succeed. If the head flit is granted in both SA and VA, the flit traverses the crossbar and is finally transferred to the next router at the LT stage. The rest of the body flits and tail flit request only SA using the same virtual channel allocated from the head flit.

However, at the routing computation stage, if the routing computation returns multiple output ports, the router considers the inserted packet a multidestination packet. The multidestination packet is handled as multiple unicast packets from the SA/VA stage, except that each flit is deleted from the input buffer after all the requested output ports are satisfied. To track the state of each forwarded multidestination flit, the input VC maintains a separate VC state per output port. As can be seen in Figure 3(a)&(d), the decoded output ports from the routing computation are latched and connected to five separate VC state units. Each VC state corresponds to the output port to which the flit needs to be forwarded and controls the pipeline stage of the forwarded packet to that output port. If the decoded output port from the routing computation produces a set of valid output ports, the corresponding VC state advances to SA/VA. In contrast, the output ports that are not valid maintain idle VC states.

Each input VC may now generate multiple output VC and switch allocation requests from multiple VC state units, but another level of arbiter in the input VC grants a single request at a time. The granted request is forwarded to the input unit until all requests from the multiple VC states are satisfied. Since each VC state has private pointers of head and tail into the input buffer, the VC state keeps track of forwarded and not-forwarded flits to the corresponding output ports. At the end of the packet, when the tail flit leaves the input buffer at the ST stage, the VC states for multicast go back to the idle state.

3.2 Dynamic Packet Fragmentation

The above described operation works fine in the absence of any blocking situation. Flits proceed in a pipelined manner without stalls in the ideal case where there is no contention. However, the router pipeline can be easily blocked in the following deadlock situation with multiple contending multicast packets.

Figure 2 shows a situation where two multicast packets are requesting the north and south output ports, but not granted both output ports at the same time. The gray shaded packet is granted the north output port but not south. The black shaded packet is granted the south output port but not north. Part of each gray and black shaded packet is replicated and forwarded to each of the granted output ports. Since the multiple VC state units maintain the separate states of the forwarded packet to each output port, the multicast packet does not necessarily need to wait until all requested output ports are granted. Once the packet is granted any output port, the replicated packet can be routed to the granted output port. After the last flit in the buffer is forwarded, the packets cannot progress any more until the rest of the requested output ports are granted (gray packet to south and black packet to north). The blocking of the contending packets leads to a deadlock situation with a resource dependency cycle. Conventional solutions include deadlock avoidance by either using large enough buffers to hold entire packets or increasing the number of VCs. However, both solutions are not ideal for an area-constrained on-chip router [1], [11].

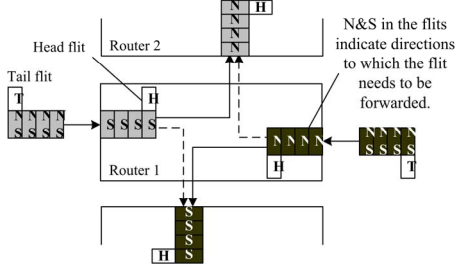


Figure 2. Deadlock

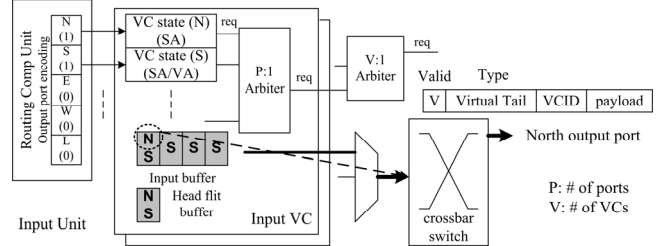
The proposed router avoids deadlock by fragmenting packets and releasing held output VCs. Figure 3 shows the process of how a packet is fragmented. Continuing the packet routing shown in Figure 2, right before deadlock happens, the west input port with the gray shaded packet produces routing requests to north and south, and only the north route is granted. The south VC state remains in the SA/VA state waiting for the south port to be granted. On the other hand, the north VC state replicates and forwards the flits until it encounters an empty buffer.

As can be seen in Figure 3(a) (input unit of west input port), even though the input buffer is already full because of the south port requested flits, the north VC state regards the buffer as nearly empty. If no other flit is coming into this input port and the north VC state is granted to forward the last available flit to the corresponding output port, packet fragmentation starts by converting the type field of the last flit to a virtual tail flit and releasing the hold of the output port (Figure 3(a)&(b)). By fragmenting the packet, deadlock is automatically resolved, breaking the cyclic resource dependency (Figure 3(b)). After fragmentation, the north VC state goes back to idle and at the same time, the south VC state of the black shaded packet also goes back to idle. Each VC state considers packet forwarding to be complete by sending a virtual tail flit.

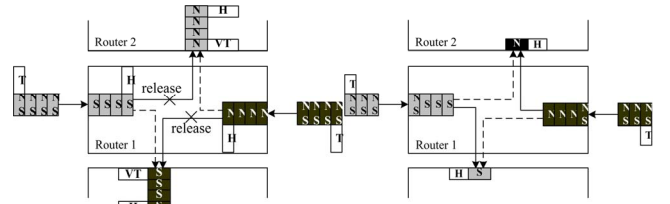
As both packets release the occupancy of output ports, the remaining flits which are waiting for requested output ports can now be granted in each requested direction (gray packet to south and black packet to north). When each packet is granted, flits are forwarded and deleted from the input buffer. From Figure 3(c), if a new flit arrives in the input buffer, the corresponding VC state of the input VC requests SA/VA again as if a new head flit arrives. The input VC stores the head flit in an extra head flit buffer when a head flit arrives at the input buffer. The buffered head flit is used for virtual head flit generation. Since the output port encodings from the routing computation unit are latched in the register at the head flit arrival and held until the tail flit leaves the input buffer, the new arrived body flit can request SA/VA based on the latched routing computation result. During the request of newly arrived flits, it is noted that the north output port is already granted to the black shaded packet, so the new flits of the gray shaded packet must wait until all north-going parts of the black shaded packet are forwarded and releases the north output port.

When north and south output ports are released, SA/VA of new flits is granted and the flits can start to be forwarded by sending the buffered head flit first (Figure 3(d)). The newly arrived body flits can not be routed by themselves since they do not contain any routing information. Therefore, the stored head flit in the head flit buffer is used to generate the virtual head flit for the fragmented packet. The fragmented packet is then reformed by

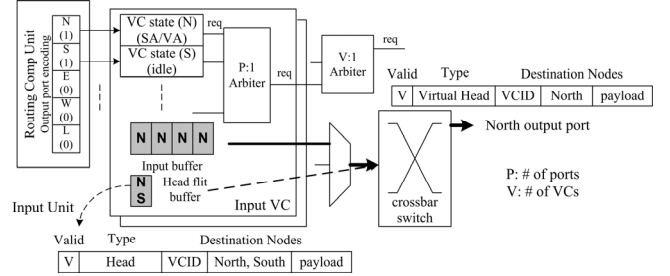
attaching the virtual head flit. The virtual head flit is copied from an original head flit in the head flit buffer, but the type field is set to virtual head flit to be distinguished from the original head flit at the destination node. With the type conversion, the destination nodes field is also modified to have nodes listed in the same direction (north). When the replicated head flit is routed, the destination nodes field is updated if necessary with correct routing information. After the virtual head flit is forwarded, the rest of the newly arrived body and tail flits simply follow the virtual head flit.



(a) Input unit of west input port with a virtual tail flit



(b) Virtual tail (VT) flit forwarding (c) New flit arrival



(d) Input unit of west input port with a virtual head flit

Figure 3. Deadlock avoidance with dynamic fragmentation

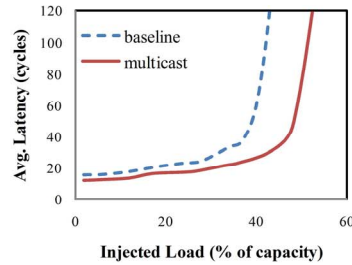
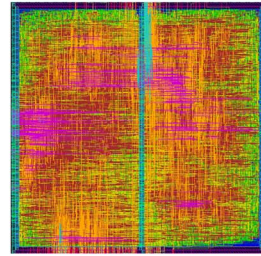
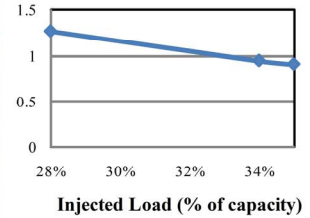
A fragmented packet can be re-fragmented if the same situation occurs in another router, so the virtual head flit overhead increases with the frequency of fragmentation. Since all fragmented packets follow the same route in deterministic routing, packet reassembly is performed by gathering the packets delivered in-order. The router is not involved in reassembly of packets, leaving packet collection to a network interface controller.

4. EVALUATION

This section describes performance and energy analyses of the implemented designs. A unicast baseline and the proposed tree-based multicast router with dynamic packet fragmentation were developed in synthesizable VHDL code. The codes were synthesized using Synopsys Design Compiler, and layouts were generated with Cadence SOC Encounter targeting the Artisan standard cell library for IBM 90 nm technology. Table I summarizes the common router features and network parameters for synthesis and circuit simulation.

Table I. Design Evaluation Parameters

Topology	Mesh 4x4
Routing	Dimension-order (XY)
# of ports	5
# of VCs	4
Buffer per port	16 (4-entry depth per VC)
Flit size	128 bits
Packet size	8-flit packet

**Figure 4. (a) Performance****(b) Multicast router layout (c) Relative energy consumption**

The performance of the proposed router is evaluated through synthetic workloads generated with uniform random traffic. The inserted network loads are a mix of unicast and multicast packets. A uniform random traffic generator adjusts overall network loads and injection sequences of multicast packets within the injected loads. For a multicast packet, the number of target destinations varies from 4 to 12 with randomly selected targets. Multicast packets are translated into multiple individual unicast packets in the baseline router case; thus, the baseline router incurs a severe serialization penalty.

Figure 4(a) shows the average packet latency versus injected loads at a 10% multicast rate. From prior literature [3], [10] assuming various multiprocessor environments, we selected a 10% multicast rate for all simulation, but maintaining coherence for large numbers of nodes in future coherence protocols may produce a much larger multicast rate. From the performance graph, the proposed multicast router outperforms the unicast baseline in terms of latency and throughput. The latency is reduced by 46% near the saturation point and 14% at zero-load. A 30% throughput improvement is also observed from the multicast routing. With multicast routing, even with fragmentation increasing the overhead with virtual head flits, the proposed router shows better performance because the number of packets in the network is reduced. To get some sense of how often fragmentation occurs, we logged the number of virtual head flits received at destination nodes. We recorded an average of 1.8 virtual head flits per multicast packet received; thus, each received multicast packet was fragmented an average of almost 2 times along its traversal. In other experiments we observed more performance benefit at higher multicast rates, as expected. The average packet latency is reduced by 53% at a 20% multicast rate at the baseline saturation point. From the presented performance result, the multicast router with dynamic packet fragmentation shows much lower latency as the multicast rate increases, so the router is very suitable for applications with high multicast rates.

Figure 4(b) shows a generated layout of the proposed router. The critical path is measured as 3.2ns including wire delays. From a layout comparison between the baseline and multicast router, the proposed design takes slightly more die area and dynamic power due to more control logic to support a multicasting capability. However, from the observed relative energy consumption in Figure 4(c), the proposed router starts to require less energy than the baseline around a 33% injection load and consumes 9% less energy at the unicast saturation point (35%). The energy crossover point would move to the left if a workload with more than a 10% multicast rate is applied.

5. CONCLUSION

This paper presented an on-chip router with hardware support for multicast using dynamic packet fragmentation. The scheme enables deadlock-free tree-based multicast routing possible since it resolves cyclical dependencies in resource allocation through packet fragmentation. The proposed router reduces latency by 38.6% and consumes 9% less energy than a unicast baseline router at the baseline saturation point. Although we introduced dynamic packet fragmentation for resolving deadlock situations in multicast scenarios, in future work we will explore how it can be used to improve performance even in unicast operations.

6. REFERENCES

- [1] R.V. Boppana et al. Resource Deadlocks and Performance of Wormhole Multicast Routing Algorithms. In *IEEE Trans. on Parallel and Distributed Systems*, Vol. 9, pp. 535-549, Jun, 1998.
- [2] X. Lin et al. Deadlock-Free Multicast Wormhole Routing in 2D Mesh Multicomputers. In *IEEE Trans. on Parallel and Distributed Systems*, Vol. 5, pp. 793-804, Aug, 1994.
- [3] M.P. Malumbres, J. Duato, and J. Torrellas. An Efficient Implementation of Tree-Based Multicast Routing for Distributed Shared-Memory Multiprocessors. In *Proc. of International Symposium on Parallel and Distributed Processing*, Oct, 1996.
- [4] Z. Lu et al. Connection-oriented Multicasting in Wormhole-switched Networks on Chip. In *Proc. Emerging VLSI Technologies and Architectures*, Mar, 2006.
- [5] A. Kumar et al. A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator in 65nm CMOS. In *Proc. of International Conference on Computer Design*, Oct, 2007.
- [6] D. R. Kumar. A New Adaptive Hardware Tree-Based Multicast Routing in K-ary N-Cubes. In *IEEE Trans. on Computers*, Vol. 50, No. 7, pp. 647-659, Jul, 2001.
- [7] C. M. Chiang and L.M. Ni. Multi-Address Encoding for Multicast. In *Proc. of the Parallel Computer Routing and Communication Workshop*, pp. 146-160, May, 1994.
- [8] J. D. Owens et al. Research Challenges for On-Chip Interconnection Networks. In *IEEE Micro*, Sep, 2007.
- [9] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Pub., 2003.
- [10] N.E. Jerger et al. Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support. In *Proc. of International Symposium on Computer Architecture*, Jun, 2008.
- [11] T. Bjerregaard and S. Mahadevan. A Survey of Research and Practices of Network-on-Chip. *ACM Computing Surveys*, Vol. 38, March, 2006.