

# Dynamic Packet Fragmentation for Increased Virtual Channel Utilization in On-Chip Routers

Young Hoon Kang, Taek-Jun Kwon, and Jeff Draper  
University of Southern California / Information Sciences Institute  
{youngkan, tjkwon, draper}@ISI.EDU

## Abstract

*Conventional packet-switched on-chip routers provide good resource sharing while minimizing latencies through various techniques. A virtual channel (VC) is allocated on a per-packet basis and held until the entire packet exits the VC buffer. This sometimes leads to inefficient use of VCs at high network loads. A blocked packet can affect adjacent routers, resulting in a congestion propagation effect. In such a scenario, VC buffers may be empty although they are regarded as fully occupied by a blocked packet. This paper proposes a dynamic packet fragmentation technique which releases empty VC buffers by fragmenting packets and allowing other packets to use the freed VC buffers. Thus, fragmentation increases VC utilization. Simulation experiments show performance improvement in terms of latency and throughput up to 20% and 7.5%, respectively.*

## 1. Introduction

Networks-on-Chip (NoCs) have been suggested as a scalable communication solution for many-core architectures. As the number of System-on-Chip (SoC) cores increases, power and latency limitations make buses increasingly unsuitable. In contrast, NoCs offer fundamental benefits of high bandwidth, low latency, low power and scalability.

Prior literature [1][4][8][11] proposed highly performance-driven router architectures with dynamic channel sharing. A packet-switched virtual channel (VC) router exploits multiple VCs, achieving high throughput with dynamic allocation of resources. A high utilization of the physical link is accomplished through multiple VCs, but little effort has focused on VC utilization. Especially given the limited number of VCs in on-chip routers, better VC utilization techniques must be considered.

As adopted in most VC routers, a VC is allocated on a per-packet basis and de-allocated at the time that the entire packet exits the VC buffer. This characteristic sometimes leads to inefficient use of VCs at high traffic loads. With a trend of shallow flit buffers per VC, a blocked packet easily propagates congestion to neighboring routers and can cause congestion to spread to the overall network. A blocked packet can cause VC buffers to be regarded as fully occupied even when they are empty. Several studies [5][12][14] characterized buffer utilization with various traffic models and proposed power optimization by placing idle buffers in a sleep mode. This paper, however, proposes a technique to exploit empty VC buffers to increase throughput and reduce latency via increased VC utilization.

We propose dynamic packet fragmentation to prevent some VC blocking scenarios from propagating congestion to adjacent routers. Once a packet is fragmented, the blocking is localized by releasing the hold of empty VC buffers, thereby allowing other packets to use the freed VC. Thus, congested upstream routers do not force inputs of downstream routers to throttle to a reduced rate. The fragmentation router, rather, provides more flexible flow control and VC utilization at high network loads.

The implemented fragmentation router is evaluated through various simulation experiments with synthetic workloads. Performance benefits are demonstrated compared to a baseline router, and accurate power and area measurements are analyzed from a layout. The result demonstrates that the fragmentation router outperforms the baseline while it consumes less energy. This result of packet latency reduction and increased throughput justifies the slightly more complex flow control required, making the fragmentation router suitable for future NoC design.

This paper is organized as follows. Section 2 describes a baseline router used for comparison in the evaluation experiments. Section 3 elaborates on the dynamic packet fragmentation technique, and the scheme is applied to the proposed model in Section 4. Section 5 discusses the simulation results, and Section 6 concludes the paper.

## 2. Baseline router

To evaluate the concept of dynamic fragmentation, we must first define a baseline router. This section discusses key attributes of a baseline router used in our evaluation experiments. A single cycle baseline router has been implemented based on the presented parameter selections, and the same baseline router is used for incorporating the dynamic fragmentation scheme.

Figure 1(a) shows the pipeline stages and steps of a baseline router. The basic steps are comprised of buffer write (BW), pre-routing computation (preRC), virtual-channel allocation (VA), switch allocation (SA), switch traversal (ST), and link traversal (LT) [3]. The following sections describe each of the basic steps adopted in the baseline router.

**BW & preRC stages:** When a flit is written into an input buffer, routing pre-computation (look-ahead routing [6]) is concurrently performed to reduce the number of pipeline stages. This pre-computation eliminates the RC stage from the critical path by overlapping it with the BW stage.

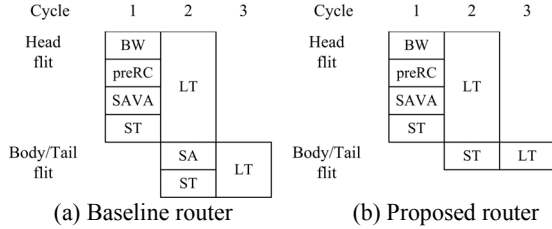


Figure 1. Pipeline stages of the routers

**SAVA stage:** With the information of the routing computation, the VA for a free VC and SA for the crossbar time slot are the next steps. Peh and Dally [1] suggested speculation techniques for improving latency. Mullins et al. [8][11] proposed low-latency routers by predetermining the arbitration decisions one cycle earlier than they are requested. In this baseline router, we followed the scheme suggested in [4]. VA and SA are done in the same cycle, but VA is performed sequentially after the SA for a combined SAVA stage. A SA winner is first selected through two separate stages of arbitration (local arbitration and global arbitration). VA is then accomplished simply by finding a free output VC from the requested output port of the SA winner.

**ST stage:** Bypassing the input buffer [7] is another technique to optimize performance. In this case, upon successful arbitration, a flit goes directly to the crossbar without delaying one cycle for input buffering. Figure 2 shows how bypassing is implemented. Since the depth of the FIFO buffers in this design is assumed to be small (i.e. 5-entry buffer), the input buffers are implemented with synthesizable flip-flops. Each entry of the flip-flop buffers can act as an input pipeline eliminating the separate input pipeline overhead. The read operation is done in parallel with local arbitration of the SA. The VC winner from the local arbitration selects a request among the VCs and makes the selected flit ready at the crossbar. While the local VC winner requests global arbitration, the flit from the input port can cross the crossbar half-way to the output port, awaiting the final arbitration result.

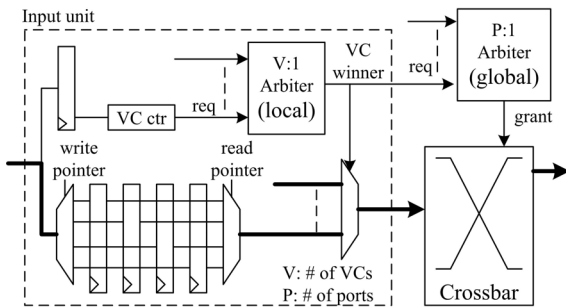


Figure 2. Bypassing in Flip-Flop based buffers

If the SA request to the global arbitration is granted and the VA succeeds by finding a free output VC, the header flit traversal is complete. At the LT stage, the header flit is finally transferred to the next router through the link. The rest of the body and tail flits use the same VC allocated to the header flit. Only the SA is performed while they traverse the crossbar as shown in Figure 1(a).

### 3. Dynamic packet fragmentation

Now that a baseline router definition has been established, this section introduces the concept of dynamic packet fragmentation to increase VC utilization. Given the trend of a small number of VCs in NoCs to minimize router overhead, VC utilization is a key factor for performance improvement. We first analyze blocking scenarios to show where poor VC utilization occurs, motivating the need for dynamic fragmentation.

The baseline router described in the previous section is adequate in the absence of blocking situations. Flits proceed in an orderly fashion through router pipelines when there is little network contention. However, at high traffic loads router pipelines often stall, propagating congestion throughout the network. Router pipeline stalls are generally classified into two types: packet stalls and flit stalls. The packet stalls are related to the packet processing functions of the pipeline whereas the flit stalls occur when a flit cannot complete the switch allocation [3]. Flit stalls can further be subdivided into credit stalls, buffer empty stalls, and stalls due to a failing SA for the switch time slot.

These flit stalls cause a packet to span multiple routers holding the VCs, as shown in Figure 3. In Figure 3(a), the blocked packet in router 3 generates a credit stall to router 2, and the credit stall propagates to upstream routers. The packet is stuck in the VCs until the credit is returned from router 3 to router 1. The situation is even worse with longer packets. Longer packets span more routers, so the chance that the flow is congested by the holding of multiple network resources is increased.

Figure 3(b) shows that mid-packet blocking in a delayed packet leads to a buffer empty stall. While packet 1 is being forwarded to router 2, router 1 may become congested due to other conditions. The flits of packet 1 that are already forwarded to router 2 before congestion occurs are routed to router 3 without any delay because router 2 is not congested. The delayed packet in router 1 causes the VCs in the downstream routers to be throttled to a reduced rate. Router 2 and 3 may experience empty buffer stalls, preventing the VC, although empty, from being assigned to packet 2. These stalls lead to a bandwidth loss and increased network congestion.

In fact, the VC router design mitigates the effect of packet blocking by sharing a physical channel among several VCs. Although the VC router design increases physical link utilization with dynamic sharing, it does not address VC utilization. Efficient utilization of the limited number of VCs can be a key factor to further performance improvement. Since the number of VCs cannot be increased without limit due to area, power and latency constraints, a better utilization technique of VCs should be considered.

Dynamic packet fragmentation provides a means for VCs to be utilized more efficiently by preventing blocking from propagating in some situations. Since the router dynamically fragments packets in these blocking situations, the frequency of fragmentation is dynamically adjusted to network traffic. Indeed, this router design avoids fragmentation overhead at low traffic rates where there is little contention. The following sections describe how fragmentation is applied to the blocking situations we address.

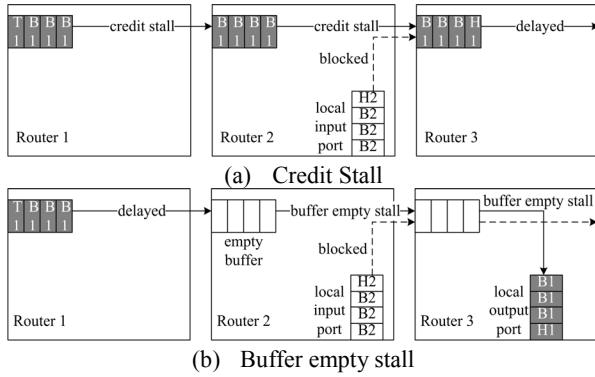


Figure 3. Examples of flit stalls

### 3.1 Fragmentation at credit stall

Since a credit stall is induced by the blocking of a downstream router, the stall is released when a flit of the downstream router leaves the input buffer and a credit is returned. The returned credit experiences latencies due to transmission and updating of the credit count. The buffer can then be assigned to a new flit. At the time of a credit stall, even though there may be idle buffers in other comparable VCs in the downstream router, the current packet cannot use the empty buffers because of VC allocation restrictions. However, if the packet is fragmented and considered as a different packet, the stalled packet of the current node can be assigned to a new VC and be routed without credit stalls.

Figure 4 shows the process of dynamic packet fragmentation in the case of a credit stall. The VC controller in the router senses the lack of credits when the VC has 1 credit left and no more credits are returned (Figure 4(a)). When the router uses the last available credit by forwarding a body flit, the VC controller fragments the packet to avoid credit stalls. The fragmentation starts by changing the type field of the sending body flit to a virtual-tail flit. The VC controller then regards the packet forwarding to be complete and releases the hold of the output VC (in this example, output VC 0 is released, Figure 4(b)).

Now, the remaining body flits in the input buffer are considered as a new packet and attempt to acquire a new VC. A copy of the header flit from the original packet must be maintained for fragmentation. This header flit copy serves as a virtual header flit for all fragments of the original packet and is treated as a normal header flit by VC controllers. With the buffered header information, the VC controller can treat the remaining body and tail flits as a new complete packet. The router performs VA and SA steps again as if a new packet is delivered. If the VA and SA succeed, the VC controller sends a virtual-header flit first using the information of the buffered header (in this example, output VC 1 is acquired, Figure 4(c)). After that, the body and tail flits follow the same route setup by the virtual-header at the success of SA. Since a new VC is allocated to the fragmented packet, the credit count is also updated to the amount of available buffers at the new VC.

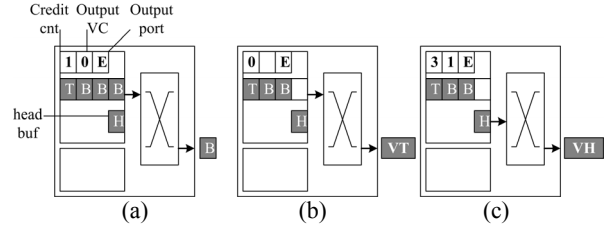


Figure 4. Packet fragmentation at credit stall

By regarding credit stalls as a congestion metric of downstream routers, a fragmented packet could be routed to a different path in adaptive routing schemes. This enables dynamic, flexible finer-grained flow control. We leave more details about coupling fragmentation with adaptive routing as a future work.

### 3.2 Fragmentation at buffer empty stall

Buffer empty stalls are caused by upstream routers, and such a stall is released when a new flit is delivered to the input buffer. A buffer empty stall prevents empty storage from being used, as shown in Figure 3(b). To give other packets a chance to use the empty buffer, packet fragmentation can be performed.

Figure 5 shows the process for how a packet is fragmented in the case of a buffer empty stall. From Figure 5(a), a router detects that it may encounter a buffer empty stall when it has a single flit in the input buffer and no flit coming into this buffer. When the last available flit in the input buffer is forwarded to the next router, the VC controller fragments the packet by changing the type field of the sending flit to a virtual-tail (Figure 5(b)). With the fragmentation, the VC controller releases the hold of the output VC (in this example, output VC 0 is released), regarding the packet forwarding as complete.

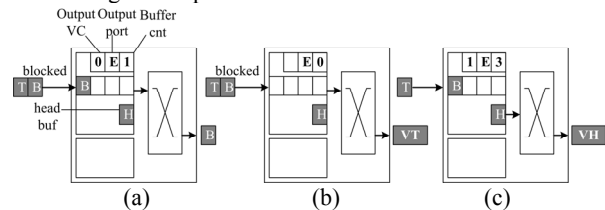


Figure 5. Packet fragmentation at buffer empty stall

When the delayed upstream router is released, and a new flit arrives in the input buffer, the VC controller treats the flit as a new packet. As described in the previous section, since the newly arrived body flit does not have any of the routing information, the VC controller retains a copy of the header flit and uses it for VA and SA steps for packet fragments.

For the routing pre-computation step, the pre-computed routing information is already available in the header buffer; thus, the routing computation process can be skipped for trailing packet fragments. If the VC controller succeeds in VA and SA, the buffered head flit is forwarded first as a virtual-header (in this example, output VC 1 is acquired, Figure 5(c)) similar to that shown for credit stall fragmentation. The rest of the newly arrived body flits follow

the same route of the virtual-header flit until the VC controller detects another buffer empty stall or tail flit.

Fragmentation prevents the mid-packet blocking from propagating to multiple routers. Figure 6 shows how fragmentation mitigates the example of Figure 3(b). Once the empty buffer in router 3 is released by fragmentation from the empty buffer of router 2, the buffer can be utilized by packet 2 as soon as the head part of the fragmented packet 1 leaves the buffer. 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.

A fragmented packet can be fragmented again if the same situation occurs in another router. So the virtual-head flit overhead is increased according to the number of times fragmentation occurs. Note that the virtual-head and virtual-tail flits are created during routing at the point of fragmentation; intermediate routers treat them identically as normal head and tail flits, respectively.

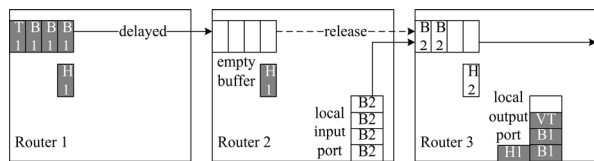


Figure 6. After the fragmentation of Figure 3(b)

#### 4. Proposed router

This section describes how dynamic packet fragmentation is applied to the baseline router described in Section 3. Although fragmentation is introduced for better resource utilization, unnecessary fragmentation should be avoided due to the overhead involved. The proposed router uses the following techniques to address this trade-off.

In the case of credit stall fragmentation, fragmentation just because of the credit loop latency needs to be avoided. The flit buffers are recycled when a credit is returned. If the number of flit buffers is below the number of cycles of the credit loop, the VC will incur a credit stall without any blocking. In this paper, a single-cycle router latency and single-cycle link latency are assumed. So, a 5-entry buffer per VC is sufficient to cover the credit loop of 5 cycles. Therefore, the router does not experience a credit stall just because of the credit loop latency.

Another fragmentation which could occur in the case of buffer empty stalls can also be avoided with the priority scheme of arbitration. With a fair arbitration scheme, the VCs interleave their flits on a single physical channel, potentially causing downstream router input buffers to be throttled even when there is no blocking. If the next downstream router is not congested, the forwarded flit may be transferred to the next router immediately. The first downstream router would then fragment the packet since it experiences a lack of flits in the input buffer.

On the other hand, a winner-take-all arbitration [3][4] for the SA solves the problem. Since this scheme allocates all the bandwidth to a single packet until it sees a tail flit or the packet is blocked, the input buffer of the downstream router does not see any empty slots between consecutive flits forwarded. Once the stream is disconnected because of the stall, the packet is fragmented and the next packet gets the priority to be forwarded until the end of the stream.

From the techniques mentioned above, the input buffers of the VC always receive a complete packet stream from header (virtual-header) to tail (virtual-tail). Even though the packet is fragmented, the fragmented packet is treated as a complete packet. Forwarding a complete packet makes it possible to emulate a wormhole router in the proposed model. In fact, body and tail flits do not require the SA step in a wormhole router. They just follow the same path set up by a header flit. The proposed router performs similarly to a wormhole router, but with the use of fragmentation it does not have the problem of mid-packet blocking due to how channels are held on a per-packet basis.

The pipeline stages of the proposed model can be seen in Figure 1(b). Unlike the baseline router, the proposed model does not require a SA stage for body and tail flits. The crossbar passage is set up by a header or virtual-header flit and is valid until the tail or virtual-tail flit traverses the crossbar. The guaranteed single hop latency for body and tail flits is achieved with the circuit switching quality of the data streaming. The scheme is beneficial in terms of power consumption since the number of activities for the SA stage is reduced to a per-packet basis.

Although the presented router can be applied to various routing algorithms, we assume dimension-order (XY) routing with a two-dimensional mesh network for the simplicity of design. Since intermediate routers forward the packet fragments based on the arrival sequence at the input port, the fragmented packets are delivered in-order with deterministic routing. The packet reassembling process at the destination is straightforward and can be left to a network interface controller.

#### 5. Simulation results

This section describes performance and power analyses of the implemented designs. A baseline and the proposed dynamic packet fragmentation routers 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 1. Design evaluation parameters

Topology	Mesh 4x4	Routing	Dimension-order (XY)
# of ports	5	Buffer per port	24 (6-entry depth per VC)
# of VCs	4		
Flit size	128 bits	Packet length	8, 16 flits

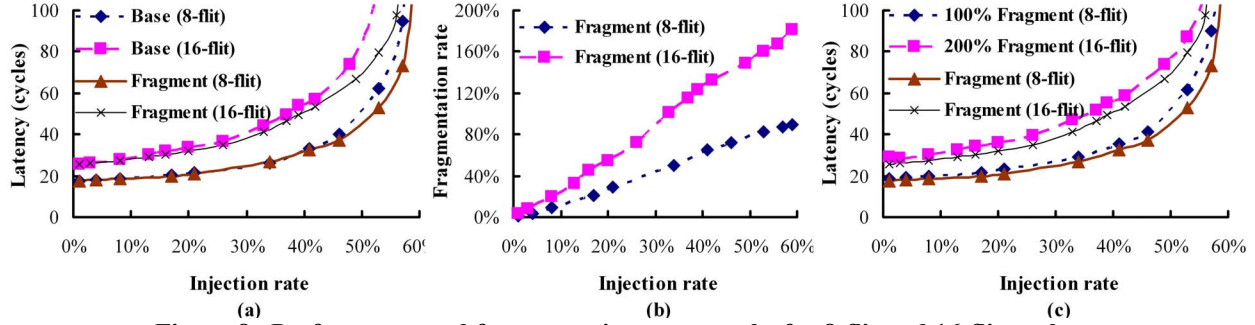


Figure 8. Performance and fragmentation rate graphs for 8-flit and 16-flit packets

## 5.1. Performance

The performance of the proposed router is evaluated in cycle-accurate simulations using synthetic workloads generated with uniform random traffic. Figure 8 shows a set of performance graphs and fragmentation rates based on various injected loads. Since the proposed router, which performs fragmentation, has an extra header buffer in addition to the 5-entry flit buffer, baseline routers are assumed to have 6-entry flit buffers for fairness of the comparison.

Figure 8 shows the simulation results for packet sizes of 8 flits and 16 flits, respectively. *Base* indicates the same baseline router mentioned in the section 3, and *Fragment* indicates the proposed router. Both routers are assumed to have a winner-take-all arbitration scheme.

As can be seen in Figure 8(a), in the 8-flit packet simulation, *Base* shows close performance to the proposed router. Since the 8-flit packet can just be buffered in two adjacent routers assuming 6-entry input buffers, the interference of the blocked packet to other packets was minimal. On the other hand, in the 16-flit packet simulation, the fragmentation router is superior to *Base* for latency and throughput. The proposed router shows 20% less latency at the *base* saturation point of a 52% injection rate and 7.5% more throughput between the respective saturation points. As expected, this result indicates that fragmentation is more beneficial to longer packets. Although fragmentation increases the overhead in terms of added virtual header flits, the increased resource utilization and dynamic reaction capability more than compensate for the added overhead.

Figure 8(b) shows the fragmentation rate of the proposed router versus the injection rate. The fragmentation rate is measured by counting the number of virtual header flits among the total number of packets received at the destination node. So, 100% and 200% fragmentation indicate that every packet is fragmented in 8-flit and 16-flit packet simulations, respectively. Note that the size of the fragmented packet is determined by the number of buffer entries. Therefore, with 6-entry input buffers the 8-flit packet is fragmented only once and the 16-flit packet is fragmented up to twice. From Figure 8(b), the fragmentation is gradually increased as more traffic is applied, as expected. At low traffic loads, most packets are delivered to destination nodes without fragmentation because blocking is less likely. At the

saturation point, however, most packets are fragmented due to congestion.

To assess the impact of the dynamic nature of the proposed fragmentation scheme, the performance of dynamic fragmentation is compared to the performance of a static fragmentation scheme where packets are fragmented at the point of injection. Figure 8(c) shows the performance graphs of 8-flit and 16-flit packets. *100% Fragment* and *200% Fragment* indicates statically fragmented packets whereas *Fragment* denotes the proposed dynamically fragmented scheme. From both graphs, dynamic fragmentation shows more than a 10% latency reduction at the saturation points of the statically fragmented packet cases. This indicates that the dynamic and reactive aspect of our proposed router is an important aspect of the design.

## 5.2. Place and Route

The *Fragment* and *Base* routers were synthesized with clock gating to minimize dynamic power. Table 2 shows the layout picture of the fragmentation router and a breakdown of the router area. This section provides accurate timing, area, power, and energy analyses based on the generated layouts.

Table 2. Fragmentation router layout & area

	<i>Base</i>	<i>Fragment</i>
Cell area	400673 $\mu\text{m}^2$	413773 $\mu\text{m}^2$
Flit buffer	72%	60%
Header	0%	10%
Crossbar	5.4%	5.5%
VC ctrl	12%	14.1%
[820x820] $\mu\text{m}^2$ with 76.73% density		

**5.2.1. Area.** Although the *Fragment* router is slightly bigger than the *Base* due to the slightly more complex control logic for fragmentation, the area difference is negligible (approximately 3%). Instead, data-path elements (flit buffers, header buffer, & crossbar) dominate over the control elements by taking over 75% of the entire area for both designs. Since the area is roughly proportional to the dynamic power, this measure is also indicative of the power consumption.

**5.2.2. Critical path.** The critical path is measured as 3.0 ns including wire delays for both designs. Fragmentation is not on the critical path. Note that the fragmentation router has a characteristic of wormhole router, unlike highly dynamic VC

allocation. So, the pre-allocation technique of resources can be easily applied to it. Although several techniques for low-latency routers were proposed in [8][11], they are quite complex and were not applied to maintain the simplicity of our design.

**5.2.3. Power.** To see the impact of traffic to the overall network power consumption, the network power consumption is measured with the same workload used in the performance evaluation. Every annotated switching activity is captured and reflected in the power calculation using Synopsys Power Compiler with typical operating conditions (1.0V, 25°C). The result of Figure 10(a) shows the power requirement according to the various traffic rates. Given that the main overhead of the *Fragment* router is due to the more complex control logic, the power overhead can be regarded as more overall switching activities due to the larger number of standard cells and utilizing empty buffers.

**5.2.4. Energy.** Power alone can be a misleading metric. The correlation of power and performance must be considered to evaluate the true merit of a design. The energy cost graphs in Figure 10(b), therefore, are a better reflection of the design efficiency. The dynamic energy per packet for the applied synthetic workloads is indicated by multiplying the latency and the network power consumption [13]. As can be seen in Figure 10(b), the *Fragment* router consumes almost the same energy as *Base* at low injection rates. However, at high injection rates near the saturation point, the *Fragment* router consumes 15% less energy than *Base*.

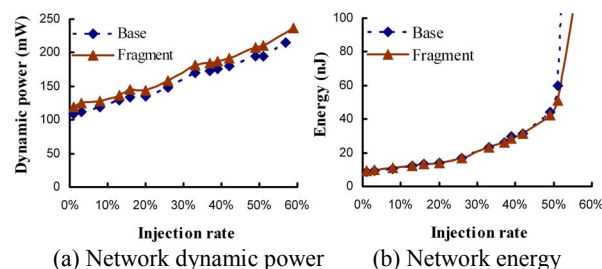


Figure 10. Network power and energy consumption

## 6. Conclusions

A dynamic packet fragmentation router has been presented and evaluated in terms of performance, power, and energy. The fragmentation router increases performance in terms of latency and throughput up to 20% and 7.5%, respectively. Moreover, simulation results indicate an energy savings as well. Since dynamic fragmentation reacts to traffic conditions, it increases VC utilization and relieves congestion. While introducing the packet fragmentation technique in on-chip routers for the first time, we believe that the presented idea opens up another research area for flow control mechanisms in NoC router design. For future work, we are exploring how the decision to fragment can be made more intelligently. Such work is necessary to scale the concept for larger packets so that the fragmentation overhead does not negate the benefit.

## 7. References

- [1] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers", In *Proceedings of International Symposium on High-Performance Computer Architecture*, Pages 255-266, January 2001.
- [2] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip", *ACM Computing Surveys*, Vol 38, March 2006.
- [3] W. J. Dally and B. Towles, "Principles and Practices of Interconnection Networks", *Morgan Kaufmann Publishers*, 2003.
- [4] A. Kumar et al, "A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator in 65nm CMOS" In *Proceedings of International Conference on Computer Design*, October 2007.
- [5] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks", In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture*, January 2003.
- [6] M. Galles, "Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER chip", *IEEE Micro*, Vol 17, p.34-39, January 1997.
- [7] H. Wang, L.-S. Peh, and S. Malik, "Power-driven Design of Router Microarchitectures in On-chip Networks" In *Proceedings of the 36th annual International Symposium on Microarchitecture*, December 2003.
- [8] R. D. Mullins, A. F. West, and S. W. Moore, "Low-Latency Virtual-Channel Routers for On-Chip Networks", In *Proceedings of the 31st International Symposium on Computer Architecture*, June 2004.
- [9] W. J. Dally, "Virtual-Channel Flow Control" In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, May 1990.
- [10] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "Resource Deadlocks and Performance of Wormhole Multicast Routing Algorithms" In *IEEE Transactions on Parallel and Distributed Systems*, Vol 9, p.535-549, June 1998.
- [11] R. D. Mullins, A. F. West, and S. W. Moore, "The Design and Implementation of a Low-Latency On-Chip Network", In *Proceedings of the 11th Asia and South Pacific Design Automation Conference*, January 2006.
- [12] X. Chen and L.-S. Peh, "Leakage Power Modeling and Optimization in Interconnection Networks", In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, August 2003.
- [13] A. Banerjee, R. Mullins, and S. Moore, "A Power and Energy Exploration of Network-on-Chip Architecture", In *Proceedings of the 1st International Symposium on Networks-on-Chip*, May 2007.
- [14] H. Matsutani et al, "Adding Slow-Silent Virtual channels for Low-Power On-Chip Networks", In *Proceedings of the 2nd International Symposium on Networks-on-Chip*, April 2008.