2012 Sixth IEEE/ACM International Symposium on Networks-on-Chip

Fine-Grained Bandwidth Adaptivity in Networks-on-Chip Using Bidirectional Channels

Robert Hesse, Jeff Nicholls and Natalie Enright Jerger
Department of Electrical and Computer Engineering, University of Toronto
Toronto, Ontario, Canada
Email: {hesserob, nichol79, enright}@eecg.toronto.edu

***Abstract*—Networks-on-Chip (NoC) serve as efficient and scalable communication substrates for many-core architectures. Currently, the bandwidth provided in NoCs is overprovisioned for their typical usage case. In real-world multi-core applications, less than 5% of channels are utilized on average. Large bandwidth resources serve to keep network latency low during periods of peak communication demands. Increasing the average channel utilization through narrower channels could improve the efficiency of NoCs in terms of area and power; however, in current NoC architectures this degrades overall system performance. Based on thorough analysis of the dynamic behaviour of real workloads, we design a novel NoC architecture that adapts to changing application demands. Our architecture uses fine-grained bandwidth-adaptive bidirectional channels to improve channel utilization without negatively affecting network latency. Running PARSEC benchmarks on a cycle-accurate full-system simulator, we show that fine-grained bandwidth adaptivity can save up to 75% of channel resources while achieving 92% of overall system performance compared to the baseline network; no performance is sacrificed in our network design configured with 50% of the channel resources used in the baseline.**

I. Introduction

To meet the growing communication demands of future many-core applications, architectures will employ networks-on-chip (NoC) to provide a scalable high-bandwidth, low-latency communication substrate. Recent research demonstrates that there is no *one size fits all* NoC architecture [12]. While chip multiprocessor (CMP) workloads behave inherently dynamically, there is little work that focuses on designing a network to accommodate and respond to this properly. For example, the bandwidth demands of the network can vary throughout the execution of a single application, yet the communication infrastructure remains a fixed, rigid design.

Prior work [12] focuses on inter-application variations and proposes a network that can adapt at a very coarse time granularity. In addition to coarse-grained changes in communication demand, we observe significant intra-application variation. Applications and traffic patterns are not fixed and vary heavily over time. A NoC implementation that provides superior communication performance for one traffic scenario might be underutilized or over-provisioned at a later point in the execution.

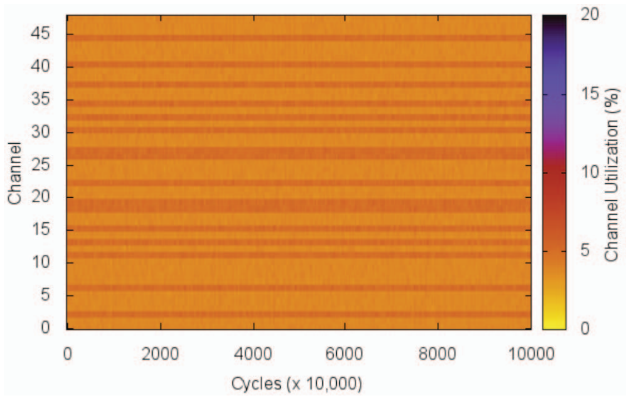


Figure 1. Channel utilization of uniform random traffic in a 4 × 4 mesh network

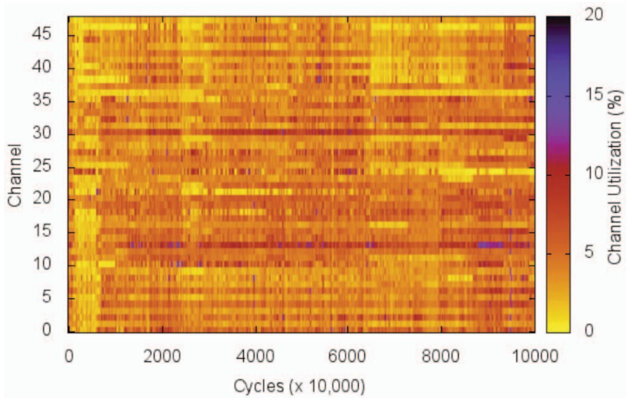


Figure 2. Channel utilization of Facesim traffic in a 4 × 4 mesh network

To efficiently utilize hardware resources and improve overall system performance, NoCs need to dynamically adapt to changing traffic patterns. Although average injection rates for NoCs are generally low, today's NoCs are designed to tolerate peak traffic loads in order to avoid becoming the performance bottleneck during periods with high volumes of communication. The bandwidth resources of such a NoC are statically provisioned at design time with only limited knowledge about application requirements. This work explores whether a dynamically adapting communication infrastructure could be designed for the average case without losing performance during phases of high traffic volume.

Previous analysis [1], [7] of different single-threaded (SPEC CPU2000 [8]) and multi-threaded workloads (SPLASH-2 [20], PARSEC [2]) for CMPs reveals very low average injection rates, generally under 5%, which is consistent with our own simulations. With such low injection rates, a dynamically adapting NoC could share resources across the network to improve performance and make the network more resistant to hardware failures.

NoC designs are often evaluated using synthetic traffic patterns, such as uniform random or tornado traffic [4]. These patterns have constant packet injection rates and regular communication patterns; they stress the network and analyze performance under known, well-structured conditions. For example, Fig. 1 shows the channel utilization for uniform random traffic over 100 million cycles on 16 cores with a 4x4 2D mesh NoC[1]. The channel utilization for each of the 48 channels is sampled periodically; since this is a synthetic traffic pattern, with a fixed injection rate, there is modest variation across channels but no temporal variation on a given channel. However, real multi-threaded CMP applications do not exhibit the same regularity. Instead they have highly dynamic communication patterns. For example, Fig. 2 shows the channel utilization for Facesim, a PARSEC application [2]. Here, we observe substantial variations both across channels and in time for a single channel. The network's performance with such dynamic, realistic traffic will differ substantially from the same network handling synthetic traffic patterns. Both temporal and spatial variations in application traffic can impact interconnect and overall system performance; NoC designs that adapt to these variations will save power and area, producing more efficient implementations.

In this paper, we explore adapting a NoC's bandwidth resources to the dynamically changing bandwidth demands of its applications. We introduce *fine-grained bandwidth-adaptivity*, which is provided by replacing the usual pair of unidirectional channels between two routers with narrow bidirectional channels. These channels can be dynamically configured in accordance with current bandwidth demands. We propose a novel router architecture to monitor the network's bandwidth demands and to switch the channels accordingly.

We target the optimization of link resources for several reasons. First, channel width has a significant impact on the router area as wide links require wider crossbars and crossbar area grows quadratically with port width. Router area can be similar in size to core area [9]; saving router area can allow designs to add additional cores, increase core complexity or add additional caches. There are complex trade-offs between these components and system performance; an area-efficient router design can improve the design of the overall system. Second, while bandwidth on chip is widely considered to be abundant, there are constraints and limitations. Among these are the ability to route large numbers of wires. It is difficult to route wires over dense logic which limits the placement and number of wires [10]. The metal layers devoted to interconnect wiring are also limited. Fewer wires can ease the layout and routing of NoCs. Finally, the wires that become available could also be used for different, performance enhancing purposes.

[1]Details about the simulation setup and methodology are in Sec. V.

Experimental evaluations on a cycle-accurate full-system simulator show that our solution significantly decreases a NoC's channel resource requirements while maintaining overall system throughput. To summarize, the main contributions of this paper are the following:

- We analyze NoC bandwidth demands of PARSEC applications and observe that average bandwidth requirements are low. We observe that bandwidth requirements change dynamically during runtime and channel resources are generally over-provisioned.
- We propose a novel, fine-grained mechanism to dynamically adapt a NoC's bandwidth to the current bandwidth demands and increase utilization without compromising network latency. We decouple flit width from channel width and thereby significantly reduce the required channel resources without incurring the increased serialization delay that normally accompanies narrow channels.
- We compare our proposal to previous adaptive bandwidth solutions [3], [13]. Our solution can decrease the channel resources of a 4x4 mesh NoC by 50% without sacrificing overall system performance and by 75% while still achieving 92% of the baseline performance.

II. Motivation

To understand the opportunities for an adaptive, reduced-resource network, we first analyze the behaviour of several CMP applications. Next, we analyze the impact of naively reducing channel width.

A. Concept of Bandwidth Adaptivity

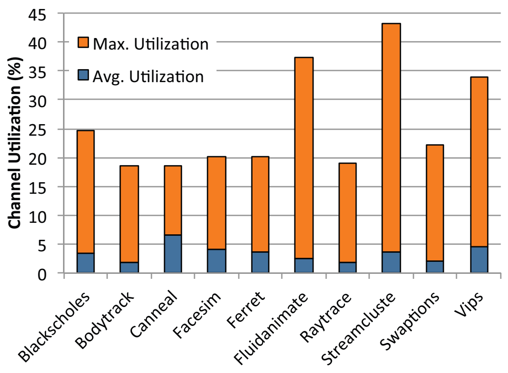
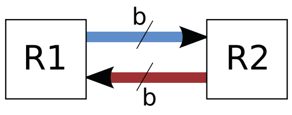
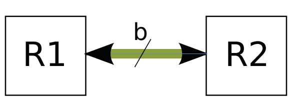


Figure 3. Channel utilization for PARSEC using the baseline 2D mesh NoC

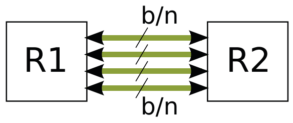
To analyze the bandwidth demands of common CMP applications, we measure the channel utilization of different PARSEC benchmarks executed on a 16-core CMP. For a period of 100 million cycles, we determine the percentage of cycles each channel in a 4x4 2D mesh NoC is utilized. Sufficient buffers and virtual channels are provided to eliminate all bottlenecks in the network; thus, maximal channel utilization can be achieved. Fig. 3 shows that the average channel utilization does not exceed 7% for any benchmark and is generally much lower. Although the maximum channel utilization is considerably higher and reaches 43% for Streamcluster, most of the time channels are idle and the network is overprovisioned in terms of its bandwidth resources. To reduce the channel resources and therefore increase their utilization, we propose to combine the data traffic of two unidirectional channels into one narrower bidirectional channel.



(a) Two unidirectional channels



(b) One bidirectional channel



(c) Four bidirectional channels

Figure 4. The concept of bandwidth adaptivity

In conventional NoCs, every two neighbouring routers are connected by two unidirectional channels of width b, as Fig. 4a illustrates. This design allows for one flit of width b bits to be sent in each direction concurrently per cycle providing a bandwidth of $2b$ bits per cycle. Since the channels are idle most of the time, it makes sense to combine their traffic on one bidirectional channel, which has the ability to transfer flits in only one direction at a time. Fig 4b shows how one bidirectional channel of width b bits can replace the two unidirectional channels providing a bandwidth of b bits per cycle. The bidirectional channel switches its direction according to the bandwidth demands of the attached routers. In order to be able to transmit data simultaneously in both directions, the bidirectional channel can be split up into a number n channels, each with width b/n bits, which can switch their direction independently; this arrangement still provides a total bandwidth of b bits per cycle, but allows for a fine-grained bandwidth-adaptive channel between two adjacent routers, as shown in Fig. 4c.

B. Advantages of Exploiting Bandwidth Adaptivity

Instead of combining the bandwidth of two unidirectional channels into bidirectional channels, one could also narrow the width of the unidirectional channels, to $b/2$ for example, as a simpler solution to increase channel utilization and save resources. Narrower channels, however, have the negative side effect of increasing the number of flits in a packet, if we assume fixed packet sizes and a flit width corresponding to the channel width. Increasing the number of flits per packet leads to a significantly worse latency across all benchmarks due to increased serialization delay and congestion in the network, as Fig. 5 demonstrates for varying channel widths in our baseline NoC. To deliver high system performance, low communication latency is imperative; 2-byte wide links incur unacceptably high latency which degrades overall performance.

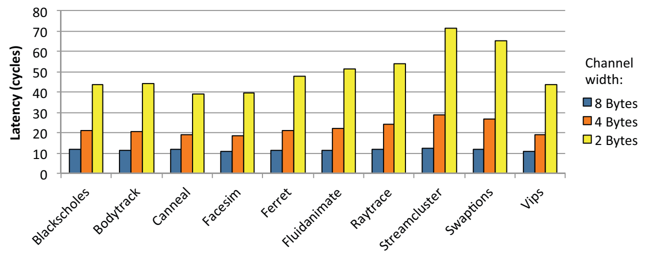


Figure 5. Average packet latency with different unidirectional channel widths

Additional delays due to increased packet length can be avoided by using bidirectional channels. Since the effective channel width per direction remains unchanged, the flit width can stay the same while the total channel resources are reduced. Therefore bidirectional channels allow for increased channel utilization without negatively affecting the length of a packet, which corresponds to greater serialization delay and latency due to congestion. However, while a bidirectional channel is transmitting a packet in one direction, another packet that might require bandwidth in the opposite direction would be stalled and additional delays would be introduced.

We mitigate this added delay by allowing the simultaneous transfer of flits in both directions through narrower channels, which independently switch their direction so that packets do not need to wait for a transmission in the opposite direction to finish. In order to transfer a flit through narrower channels without changing its width, we divide a flit into smaller subunits (phits) of width b/n for transmission across the channel. This is called *phit-serial communication* and is explained in more detail in Sec. III-A.

III. Bandwidth-Adaptive Channel Architecture

In this section, we present our modifications to the router architecture to support fine-grained bandwidth adaptivity. In a standard virtual channel (VC) router, every output port is connected to an input buffer of a neighbouring router using a unidirectional channel. In our solution, the two unidirectional channels are combined into a set of bidirectional channels that can independently be driven from either one of the adjacent routers. Control logic and tristate

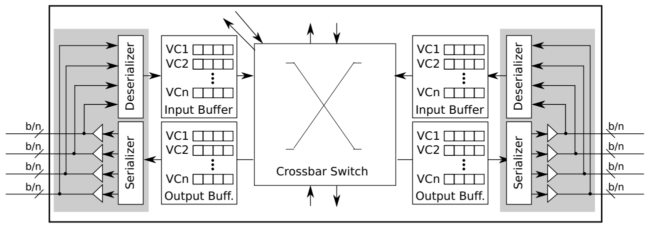


Figure 6. Modifications to standard VC router

buffers ensure that no channel is driven from both sides simultaneously. The necessary modifications to a standard VC router are shown in gray in Fig. 6 and will be discussed in the following sections. The remaining modules of the router, including allocation and flow control are unchanged.

A. Phit-Serial Communication

Conventionally, the width of a flit is considered to be identical to the channel width in NoCs. This allows for the transmission of exactly one flit per channel every clock cycle. Therefore, the bandwidth between two adjacent routers is fixed to one flit in each direction per cycle. Because the bandwidth demands of a NoC change dynamically throughout application execution, a replacement of the fixed bandwidth structure with a more flexible solution is necessary. Previous approaches [13], [3] have introduced flexible bandwidth allocation by adding more channels between routers, which enable the potential transmission of multiple flits per cycle. These designs add significant resources to the baseline architecture. Since channel width is directly coupled to the flit width, a reduction of the over-provisioned bandwidth resources in these cases is only possible if the flit width is reduced at the same time. However, reducing the flit width leads to increased packet lengths and therefore increases the packet latency.

To provide bandwidth flexibility without increasing channel resources or harming communication performance by increasing packet serialization latency, we decouple the flit width from the channel width. Flits are subdivided into smaller units called *phits* (**ph**ysical transfer un**it**)[2]. One phit represents the amount of data that can be transferred between two routers on one channel in one cycle. Flow control (e.g. buffer allocation) between routers is still managed at the granularity of a flit, which allows us to change the channel width independently of flit and buffer widths. Hence, we can reduce the channel width while keeping the packet length constant.

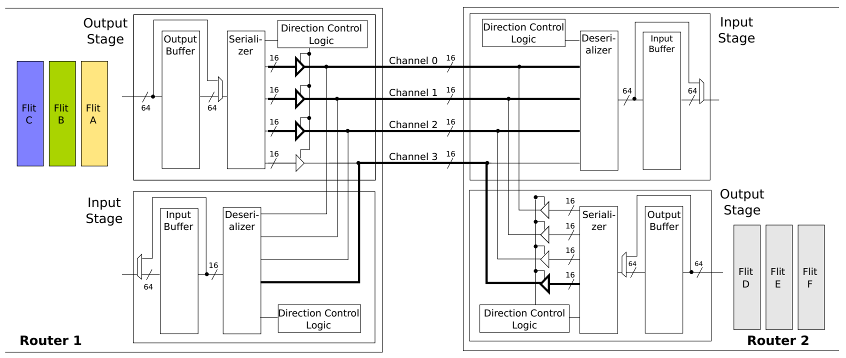
Each flit is broken down into multiple phits; phits are transferred sequentially across a narrow channel and reassembled on the receiving end. To reduce the serialization delay across the channel, multiple narrow channels can be used in parallel. The key difference between phits used in off-chip networks and our design is that multiple phits can proceed in parallel between two adjacent routers. Every channel is bidirectional and able to switch direction independently; bandwidth-adaptivity at the granularity of single phits is provided.

[2]Phits are common in off-chip networks which are often constrained by limited pin bandwidth.

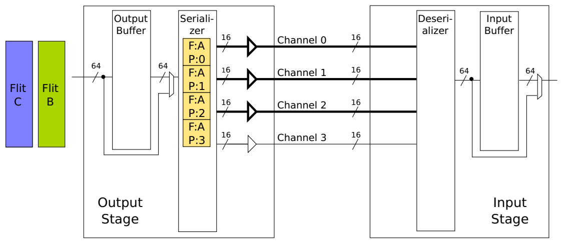
Fig. 7 walks through an example of phit-serial communication between two adjacent routers. While Fig. 7a shows the entire bidirectional input and output port setup, subsequent figures only show communication in one direction for brevity. The figures only show the input stages (IS) and output stages (OS) of our design. Three 8-byte flits (Flit A, B, and C) are divided into four 2-byte phits each, which are transferred across four bidirectional 2-byte channels. The transmission begins with an empty OS and IS; initially, the channels are configured with 3 channels pointing to the right and 1 pointing to the left (3:1) as shown in Fig. 7a. During the next clock cycle (Fig. 7b), Flit A arrives at the OS and the serializer divides it into four 2-byte phits (labeled *F:A, P:x* where x is the phit number) after bypassing the output buffer. Next, three of Flit A's phits are transferred across the three available channels to the deserializer of the downstream router where they are temporarily stored until the flit is reassembled. The remaining phit of A, (F:A, P:3) is shifted up in the serializer. At the same time, Flit B enters the OS and is written directly into the serializer as shown in Fig. 7c. The channel configuration changes during the following clock cycle (2:2), so that only two phits can be transferred to the downstream router (Fig. 7d). Transferring the last phit of Flit A completes its reassembly in the deserializer, so it leaves the IS at cycle 4 (Fig. 7e).

The proposed mechanism is able to adjust the number of channels in a particular direction on a per-cycle basis, while it guarantees in-order delivery of phits as well as flits without introducing additional delays aside from a potential serialization delay, which occurs whenever the available bandwidth is smaller than the flit width. The key to avoiding additional delays due to gaps between phits in the serializer and deserializer modules is a proper alignment of the phits towards the available bidirectional channels, while the number of transferred phits can change every cycle. When phits are allocated less channels, the packet will experience extra serialization; however as allocation is based on demand, this extra serialization does not impact performance. We avoid additional delays due to buffering by bypassing input and output buffers whenever they are empty. Output buffers are only used when the serializer is full. Fig. 8 demonstrates a low-overhead hardware implementation of our proposed phit-serial communication concept in which the proper phit alignment is accomplished by a barrel shifter in conjunction with an adjustable shift register.

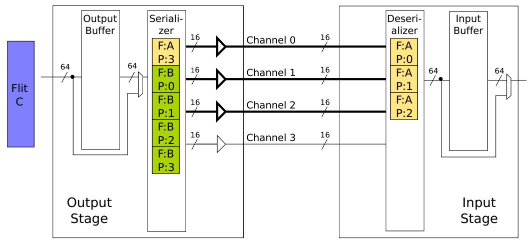
In our example, there can be a varying number of active channels at any given time. Whenever phits are present in one of the registers that are attached to an active channel,



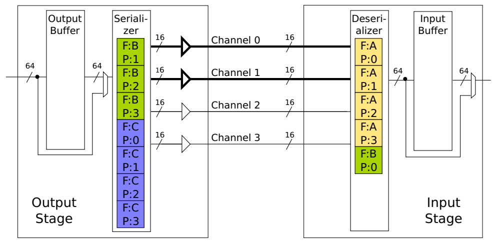
(a) Clock cycle: 0; 3 channels to the right; 1 channel to the left



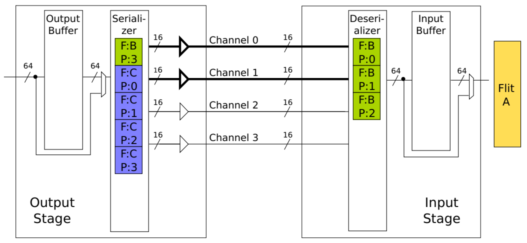
(b) Clock cycle: 1; 3 channels to right; 1 channel to left



(c) Clock cycle: 2; 3 channels to right; 1 channel to left



(d) Clock cycle: 3; 2 channels to right; 2 channel to left



(e) Clock cycle: 4; 2 channels to right; 2 channel to left

Figure 7. Concept of phit-serial transmission between OS and IS

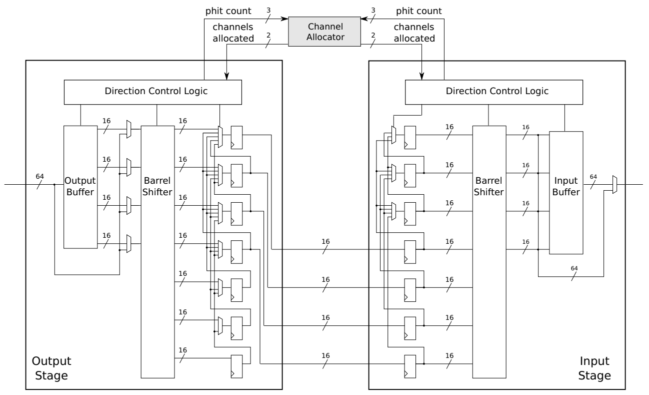


Figure 8. Implementation of the adaptive serializer/deserializer

they are transferred to the IS of the downstream router. In order for the input registers to accept the new phits, any phits that were received during the previous cycle need to be shifted out of the registers that are connected to one of the channels. If all phits of an entire flit have been assembled in the shift register, the reassembled flit can be transferred into the input buffer. Because the reassembled flit can be at any position within the shift register, we need a barrel shifter to align the flit properly with the input buffer. In the meantime, the shift register on the sending side shifts forward by the exact number of phits that were sent out during the previous cycle, so that the registers, which are attached to the channels are always supplied with new phits as long as phits remain within the OS. As soon as an entire flit fits into the shift register, the next flit is read from the output buffer, aligned through the barrel shifter and written into the shift register.

Our proposed bandwidth-adaptive serializer/deserializer mechanism is able to manage an arbitrary change in the amount of available bandwidth on a cycle-per-cycle basis, while avoiding additional delays through proper phit alignment and buffer bypassing. This allows for an efficient and low-overhead solution to decouple flit width from channel width and adjust the bandwidth between every two adjacent routers in a fine-grained manner. The overhead of our solution will be evaluated in Sec. V-D.

B. Bandwidth Allocation

Each set of bidirectional channels connecting two adjacent routers contains a channel allocator module, positioned between the two routers as shown in Fig. 8. This module controls the direction of each individual channel by sending a control signal to the direction control logic in each router port. In order to generate this control signal, the allocator module utilizes a pressure-based control mechanism which takes as input the number of phits waiting to use either output port. The phit count is sent as sideband signal halfway across the channel.

The allocator module then determines the next cycle channel allocation. The number of channels allocated in each direction depends on the phit count ratio between the two routers. The larger the difference between the phit counts, the more bandwidth allocated in one direction. In our example with 4 bidirectional channels, 2 channels are allocated in each direction if the two phit counts are equal. If one phit count is larger while the other is non-zero, 3 channels are allocated. All 4 channels are allocated in one direction only if one output port contains phits while the other does not; this policy is required for deadlock avoidance (Sec. III-C).

We compared this pressure-based approach to different metrics, such as the ratio of credit counts between two routers and the ratio of sent and received flits, but found our solution to be superior, since it directly expresses the bandwidth requirements of a particular connection.

As described in Sec V, we implement the channel allocator module without affecting the critical path of the router. By removing bandwidth allocation from the critical path, we can adjust the channel configuration on a cycle-by-cycle basis, which allows us to match the bandwidth demands of a given traffic pattern.

C. Deadlock Avoidance

The introduction of bidirectional channels can theoretically cause network deadlock due to loss of connectivity; this type of deadlock cannot be prevented using a deadlock-free routing algorithm, such as DOR. Since bidirectional channels can potentially cause connections to disappear temporarily during runtime, the network might end up not being fully connected at all times. To prevent deadlock, we need to provide full connectivity when needed. Whenever at least one phit is ready to be sent to a downstream router, one bidirectional channel will be reserved in the respective direction. The channel cannot switch its direction as long as there are phits available in the OS. This mechanism ensures that every flit in the network can progress to the next router, due to adaptively providing full connectivity. The benefit of this adaptive full connectivity is that the entire bandwidth of all channels can still be allocated in one direction provided that no phits are ready for transmission in the opposite direction.

IV. Qualitative Comparison of Bandwidth-Adaptive Designs

Although bidirectional channels have been previously proposed [3], [13], our solution represents a novel architecture with the objective of reducing resources that compliments this design space. In this section, we discuss the qualitative differences between the concepts and show why our proposal presents a significant improvement over both previous approaches. We present quantitative comparisons in Sec. V.

BiNoC [13] introduces a bidirectional channel mechanism, which consists of exactly two bidirectional links between adjacent routers instead of the standard two-unidirectional-links implementation. Since only two bidirectional links are available between each pair of routers and direction switching involves a significant delay, only some coarse-grained bandwidth adaptivity is provided. Moreover, the ability to switch the direction of channels may lead to deadlock, which the paper does not address.

A bandwidth-adaptive implementation by Cho et al. [3] refines the idea of bidirectional channels by using fine-grained adaptive, pressure-based control of multiple channels. This design is able to hide BiNoC's direction-switching delays and deals with the additional deadlock possibilities. However, this additional flexibility comes at the cost of significant area and timing overheads. Large crossbars and complicated arbitration logic are necessary to support multiple channels per port, which are all connected to the crossbar. Our solution mitigates the additional overhead by keeping the number of crossbar ports stable and solely adjusting the bandwidth distribution between adjacent routers.

Both papers follow a common strategy of increasing bandwidth capacity; BiNoC [13] provides up to two times the baseline bandwidth in one direction. Cho et al. [3] increase the number of physical channels; multiples of the baseline bandwidth are available for sending in each direction. This strategy is questionable given our observation that bandwidth resources are mostly underutilized in real-world applications and thus additional bandwidth resources do not improve application performance. Our efforts, instead, are focused on reducing NoC bandwidth resources.

V. Evaluation

We evaluate our design of a bandwidth-adaptive router (BAR) using both synthetic and real workloads, comparing it to a typical virtual-channel router (STANDARD), a BiNoC router [13] (BINOC), and an existing bandwidth-adaptive router design [3] (BWADAPTIVE).

We use FeS2 [16] for full-system x86 simulation with BookSim [4] for a detailed, cycle-accurate NoC model. We execute all PARSEC benchmarks with 16 threads on a 16-core CMP, consisting of Intel Pentium 4-like CPUs, using the *simmedium* input set, executed for 100 million cycles within the regions of interest (ROI). The system runs Fedora

Table I
SIMULATION PARAMETERS

# of Cores	16
L1 Cache (D & I)	private, 4-way, 32KB each, 64 Byte Blocks
L2 Cache	private, 8-way, 512KB each, 64 Byte Blocks
Cache Coherence	MOESI distributed directory
Network	4x4 2D-Mesh, DOR Routing

Table II
ROUTER CONFIGURATIONS FOR OUR EXPERIMENTS

Architecture	STANDARD	BINOC	BWADAPT.	BAR
Total # Buf.	5	10	20	10
Total Channels	5-in 5-out	10-inout	20-inout	20-inout
Each Buf. Size	32 flits	16 flits	8 flits	16 flits
Total Buf. Size	160 flits	160 flits	160 flits	160 flits
Crossbar	5x5	10x10	20x20	5x5

Core Linux and each thread is explicitly pinned to one of the cores to eliminate thread scheduling side effects.

Table I gives the simulation configuration. Each core has private, inclusive L1 and L2 caches with a MOESI directory protocol. We use a 4x4 2D mesh NoC for communication; however, our solution is applicable to any topology that is based on bidirectional data transfer between neighbouring nodes. We use a 2D mesh because of its simplicity and popularity within the research community. In our baseline NoC, we use a standard virtual-channel (VC) router [18] with 2 pipeline stages, lookahead routing, speculative switch allocation, 2 VCs per port and a baseline unidirectional channel width of 8 bytes.

A. Area Comparison

We use ORION2.0 [11] to compare the area of STANDARD, BINOC, and BWADAPTIVE to our bandwidth-adaptive router (BAR) for a 45 nm technology process. All routers are modelled using equal buffer resources and identical flit widths of 8 bytes. BWADAPTIVE and BAR use 4 bidirectional channels. Table II summarizes the router configurations used for comparison and Fig. 9 shows a breakdown of the area for each router.

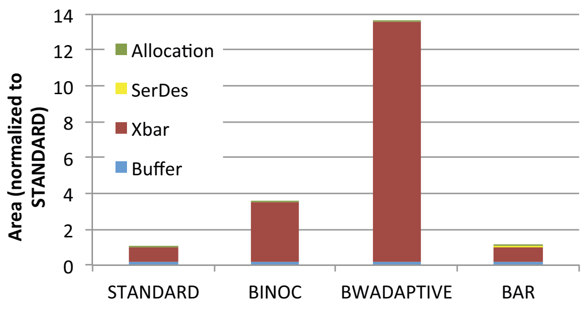


Figure 9. Router area comparison

The area for both BINOC and BWADAPTIVE is significantly larger than that of STANDARD and BAR. Both designs increase the bandwidth adaptivity of the network by providing additional channels which leads to significant area penalties. Next, we equalize the area of all routers to provide a fair performance comparison. We reduce the flit size to 4 bytes for BINOC and 2 bytes for BWADAPTIVE for all further experiments. The area results for these configurations are shown in Fig. 10. It should be noted that our adaptive serializer/deserializer (SerDes) architecture does not add significant area to the typical VC router, which was an important criteria for our design. A detailed analysis of the implementation overhead is presented in Sec. V-D.

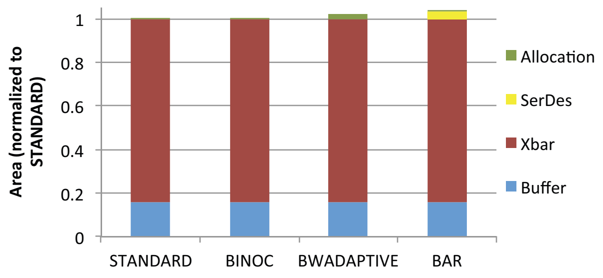


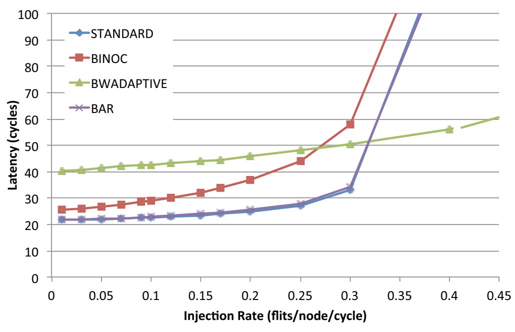
Figure 10. Router area with reduced flit size for BINOC and BWADAPTIVE

B. Network Performance Comparison

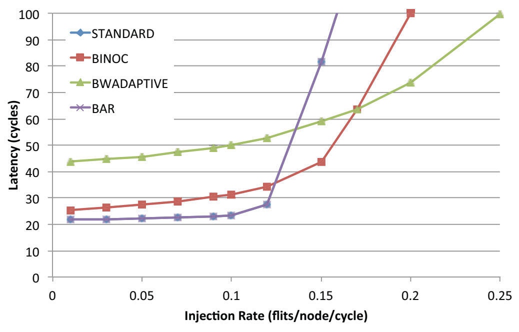
In Fig. 11, we compare the latency and throughput under uniform random, transpose, shuffle and bit-complement traffic for the 4 router designs in a 8x8 mesh with equalized areas. We simulate a 8x8 mesh to demonstrate the scalability of our design to network sizes larger than 4x4. Furthermore, we use a packet size of 32 bytes, since we measured this to be the average packet size across all PARSEC benchmarks. For synthetic traffic without temporal variation in bandwidth demands, BAR performs comparably to STANDARD. These routers have identical steady-state latency due to equally-sized crossbar and buffers. BINOC and BWADAPTIVE improve the saturation throughput over STANDARD due to more available channels, especially for highly unbalanced traffic patterns like transpose and shuffle. Both suffer from increased serialization delays caused by narrower channels resulting in higher zero-load latency. BAR keeps serialization latency low because the flit size and therefore the packet size remains unchanged. At low injection rates, which are the common operating point for NoCs, BAR outperforms both other bandwidth-adaptive designs.

C. Overall System Performance

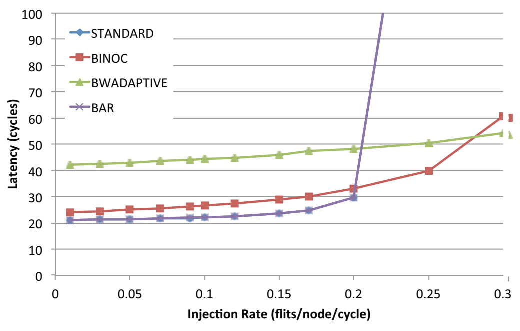
Synthetic traffic patterns fail to demonstrate the full flexibility of BAR. Therefore, we measure the overall system performance of PARSEC to gain deeper insight into the



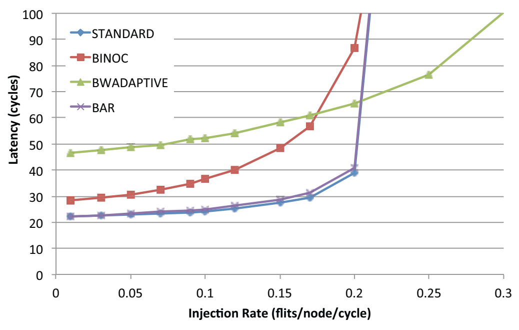
(a) Uniform Random



(b) Transpose



(c) Shuffle



(d) Bit-Complement

Figure 11. Latency-throughput graphs under synthetic traffic

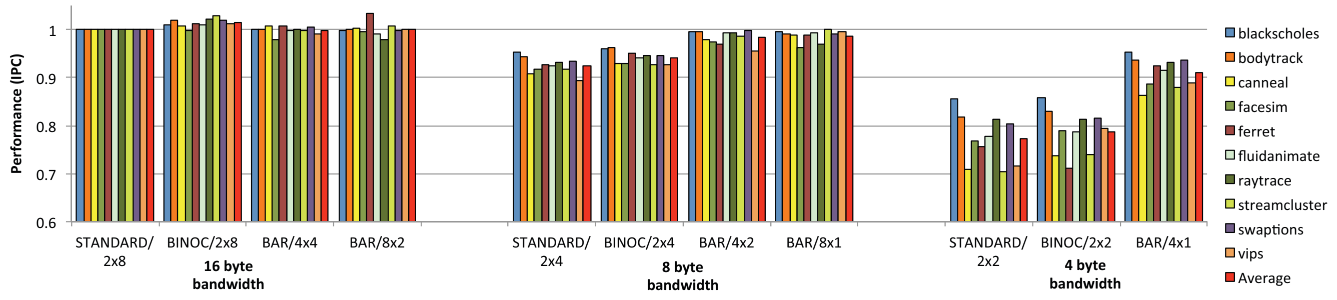


Figure 12. Performance of PARSEC benchmarks with different channel configurations

dynamic real-world behaviour of our fine-grained bandwidth adaptivity. Fig. 12 presents the normalized average instructions per cycle (IPC) across all 16 cores as a metric for application performance. All results are normalized to a STANDARD router with 8 byte unidirectional links. Each group of bars represents a different total channel bandwidth between two adjacent routers. For a standard VC router with two 4-byte unidirectional channels (STANDARD/2x4) the total bandwidth would be 8 bytes. BAR with four 4-byte bidirectional channels (BAR/4x4) has 16 bytes of total bandwidth.

The performance of STANDARD and BINOC decreases as the channel resources are reduced. Halving the channel width results in ∼10% loss in performance. Using a quarter of the original bandwidth results in more than 20% performance loss on average with a maximum loss of 30% for 3 of the workloads: canneal, streamcluster and vips. However, BAR has almost no reduction in application performance (99.3% on average) when the bandwidth is decreased by half (8 bytes). With only 4 bytes of bandwidth between adjacent routers (a 75% reduction in channel resources), BAR still achieves an average of 92% of the original performance. Increased average network latency for STANDARD and BINOC as bandwidth is reduced accounts for the performance difference. Increased serialization and congestion leads to an average latency of 50 cycles for STANDARD and BINOC, while BAR has a relatively low 20-cycle latency for the 4-byte configuration. BAR increases the average channel utilization by $4\times$, from about 3% to more than 12%.

In terms of the number of bidirectional channels, some applications benefit from a more fine-grained channel configuration. However, on average there is no significant performance difference between 4 bidirectional channels and 8 bidirectional channels. Since more channels incur a slightly higher overhead regarding the number of phits to be handled and the number of channels to be arbitrated, the configuration with fewer channels is preferable.

D. Hardware Implementation

Table III
BAR AND STANDARD POWER AND AREA

Router	Area (μm^2)	Power (mW)	Crit. path (ns)	Wires
STANDARD	316194	17.31	1.1	640
BAR	319358	17.63	1.1	345

To ensure the feasibility of our design and analyze its area and power overhead, we implement a 5-port BAR router with 4 bidirectional 2-byte channels per port, as well as a typical VC router with 5 ports and 2 unidirectional 8-byte channels per port in Verilog. Table III reports results from the Synopsys Design Compiler with a TSMC 65nm standard library; these results show that fine-grained bandwidth adaptivity can be implemented without impacting cost or performance. The additional buffer requirements for the serializer/deserializer, as well as the control logic and tristate buffers add a negligible 1% area overhead. The same is true for the routers' power consumption, which was averaged over several PARSEC application traces. While we do not incur a significant area or power overhead, we reduce the overall amount of wiring resources by 46%. The channel bandwidth is reduced by 50% while 5 bits of sideband signal information are necessary for the channel allocator module (3 bits: binary encoded number of phits in shift register; 2 bits: channel configuration).

To guarantee that our design does not increase the critical path delay of the router, we separate the task of the channel allocator module into two separate cycles. First, the phit counts are sent to the allocator and their ratio is computed. During the following cycle, the current channel configuration is sent back to the router ports and the channel direction is switched accordingly using tristate buffers. This design avoids the delay and complexity of handshaking between both router ports, such that the critical path delay can be maintained at 1.1ns.

Repeaters are necessary for long wires, which require additional control signals to switch their direction. To avoid this additional overhead, bidirectional channels are best used for local interconnects of nearby routers, such as neighbouring routers in a 2D mesh, rather than global interconnects.

VI. Related Work

Dynamically varying network requirements: Most NoC research assumes a static network architecture that is configured once at design time based on worst-case traffic analysis and does not adapt to dynamic changes in network traffic. Recently, the idea of adapting the NoC to changing workload properties has gained in popularity as it promises significant performance improvements and enhanced resource utilization.

Based on the observation that no single fixed-design NoC efficiently handles different styles of traffic, Kim et al. introduce Polymorphic On-Chip Networks [12], a special polymorphic fabric that offers per-application network customization. However, we believe the per-application granularity overlooks significant opportunity to adapt to intra-application variations. Fine-grained approaches such as ViChaR [17] target dynamic adaptation of routers by providing unified buffer resources across their input ports. By utilizing buffer space more efficiently, they achieve similar network performance using 50% fewer buffers compared to the baseline. Our work also reduces network resources through sharing of underutilized resources.

Bandwidth-adaptivity: Different approaches have been proposed to dynamically adjust a NoC's bandwidth resources to match the varying workload requirements. We present a detailed comparison between our solution and the two most closely related proposals [3], [13] in Sec. IV. These two proposals follow a completely different strategy of providing more bandwidth resources instead of reducing the already over-provisioned resources. The Adaptive Physical Channel Regulator (APCR) which allows concurrent transmission of multiple flits per channel also increases channel width [19]. Das et al. [5] and Mishra et al. [15] describe mechanisms to decouple flit width from channel size. Their approaches combine two flits when possible and then send the combined flit together through a wider channel. Like us, Meng et al. [14] observe that NoC bandwidth resources are generally over-provisioned. In order to optimize the energy-performance trade-off across varying bandwidth requirements, they dynamically adjust the channel width across the entire network. All channels are adjusted to the same width. Our approach is more fine-grained since bandwidth resources can be adjusted on a per-router basis.

Phit-serial communication: Phit-serial communication in NoCs enables better routability due to fewer wires and higher clock frequency. Dziurzanski et al. use phit-serial communication for their lossless compression system [6]. All prior uses of phit-serial communication are based on unidirectional channels and rely on the transmission of a single phit per cycle. Therefore, they implement purely serial communication. We propose a hybrid approach, where we benefit from the reduced data width of phits compared to flits, but are able to adapt the parallelism, and therefore the bandwidth of a communication link, at the same time. This allows us to reduce channel width independent of flit width without increasing packet lengths.

VII. Conclusion

In this work, we introduce fine-grained bandwidth adaptivity in NoCs using bidirectional channels. By decoupling the flit width from the channel width and utilizing a novel bidirectional phit-serial communication mechanism between routers, we are able to improve NoC channel utilization without significantly increasing communication latency. Using a full-system simulator with a cycle-accurate NoC model to execute the PARSEC benchmark suite on a 16-core CMP, we achieve a 50% reduction in channel resources for a 4x4 mesh NoC without a measurable impact on overall system performance. When reducing the channel resources by 75%, fine-grained bandwidth adaptivity is able to achieve 92% of overall system performance compared to the baseline network.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive feedback on this work. We also thank the members of Prof. Enright Jerger's research group for their helpful suggestions. This work is supported in part by the University of Toronto, NSERC of Canada and a Connaught New Researcher Award.

References

[1] N. Barrow-Williams, C. Fensch, and S. Moore, "A Communication Characterization of SPLASH-2 and PARSEC," in *Proc. of the 2009 International Symposium on Workload Characterization*, Oct 2009. 2

[2] C. Bienia *et al.*, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Proc. of the 17th Intl Conf on Parallel Architectures and Compilation Techniques*, Oct. 2008. 2

[3] M. H. Cho *et al.*, "Oblivious Routing in On-Chip Bandwidth-Adaptive Networks," in *Proc. of the 18th Intl Conf on Parallel Architecturess and Compilation Techniques*, 2009, pp. 181–190. 2, 4, 6, 9

[4] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2003. 2, 6

[5] R. Das *et al.*, "Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs," in *Proc. of the 15th Intl Symp on High Performance Computer Architecture*, feb. 2009, pp. 175 –186. 9

[6] P. Dziurzanski *et al.*, "A Lossless Compression System Realization Utilizing Phit-Serial Network-On-Chip Paradigm," in *Proc. of the 2007 European Signal Processing Conference*, Sept 2007. 9

[7] P. Gratz and S. W. Keckler, "Realistic Workload Characterization and Analysis for Networks-on-Chip Design," in *4th Workshop on Chip Multiprocessor Memory Systems and Interconnects*, 2010. 2

[8] J. Henning, "SPEC CPU2000: measuring CPU performance in the new millennium," *Computer*, vol. 33, no. 7, pp. 28 –35, July 2000. 2

[9] J. Howard *et al.*, "A 48-core IA-32 processor in 45 nm cmos using on-die message-passing and DVFS for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, January 2011. 2

[10] D. N. Jayasimha, B. Zafar, and Y. Hoskote, "On-chip interconnection networks: Why they are different and how to compare them," http://blogs.intel.com/research/terascale/ODI_why-different.pdf, Intel Corp, Tech. Rep., 2006. 2

[11] A. Kahng *et al.*, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *Design, Automation Test in Europe Conference*, April 2009, pp. 423 –428. 7

[12] M. M. Kim *et al.*, "Polymorphic On-Chip Networks," in *Proc. of ISCA*, 2008, pp. 101–112. 1, 9

[13] Y.-C. Lan *et al.*, "BiNoC: A bidirectional NoC architecture with dynamic self-reconfigurable channel," in *Proc. of the 3rd Intl Symp on Networks-on-Chip*, 2009, pp. 266–275. 2, 4, 6, 9

[14] J. Meng *et al.*, "Run-time energy management of manycore systems through reconfigurable interconnects," in *Proc. of the 21st Great lakes symposium on VLSI*, 2011. 9

[15] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, "A case for heterogeneous on-chip interconnects for CMPs," in *Proc. of the 38th Intl Symposium on Computer architecture*, 2011, pp. 389–400. 9

[16] N. Neelakantam *et al.*, "FeS2: A Full-system Execution-driven Simulator for x86," Poster presented at ASPLOS 2008, 2008. 6

[17] C. A. Nicopoulos *et al.*, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *Intl Symp on Microarchitecture*, 2006, pp. 333–346. 9

[18] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proc. of the 7th International Symposium on High-Performance Computer Architecture*, 2001, p. 255. 7

[19] L. Wang *et al.*, "An Adaptive Physical Channel Regulator for High Performance and Low Power Network-On-Chip Routers," Technical Report, 2010. 9

[20] S. C. Woo *et al.*, "The SPLASH-2 programs: characterization and methodological considerations," in *Proc. of the 22nd ISCA*, 1995, pp. 24–36. 2