Stream Arbitration: Towards Efficient Bandwidth Utilization for Emerging On-Chip Interconnects

CHUNHUA XIAO, Beijing University of Technology
M-C. FRANK CHANG, JASON CONG, and MICHAEL GILL, University of California, Los Angels
ZHANGQIN HUANG, Beijing University of Technology
CHUNYUE LIU, GLENN REINMAN, and HAO WU, University of California, Los Angels

Alternative interconnects are attractive for scaling on-chip communication bandwidth in a power-efficient manner. However, efficient utilization of the bandwidth provided by these emerging interconnects still remains an open problem due to the spatial and temporal communication heterogeneity. In this article, a *Stream Arbitration* scheme is proposed, where at runtime any source can compete for any communication channel of the interconnect to talk to any destination. We apply stream arbitration to radio frequency interconnect (RF-I). Experimental results show that compared to the representative token arbitration scheme, stream arbitration can provide an average 20% performance improvement and 12% power reduction.

Categories and Subject Descriptors: C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures (Multiprocessors)—*Interconnection architectures*

General Terms: Design, Performance

Additional Key Words and Phrases: Arbitration, Network-on-Chip, radio frequency interconnect

ACM Reference Format:
Xiao, C., Chang, M-C. F., Cong, J., Gill, M., Huang, Z., Liu, C., and Reinman, G., and Wu, H. 2013. Stream arbitration: Towards efficient bandwidth utilization for emerging on-chip interconnects. ACM Trans. Architec. Code Optim. 9, 4, Article 60 (January 2013), 27 pages.
DOI = 10.1145/2400682.2400719 http://doi.acm.org/10.1145/2400682.2400719

1. INTRODUCTION

As we enter the era of many-core and beyond, the number of cores, coprocessors, and on-chip accelerators grows rapidly. The dramatic increase of these processing elements (PE) imposes a tremendous bandwidth requirement on the communication to the memory/cache [Kumar et al. 2005]. This communication is typically accommodated via an on-chip interconnection network (or NoC: Network-on-Chip). It has been observed that electronic networks cannot efficiently supply the dramatically increased PE-memory communication bandwidth due to unacceptable power and area consumption [Kumar et al. 2005]. Therefore, alternative interconnects—such as Radio Frequency Interconnect (or RF-I) [Chang et al. 2008c], and optical interconnect [Vantrease et al. 2008, 2009]—have become more attractive as a means to scale bandwidth and latency in

This research was supported by the NSF Expeditions in Computing Award CCF-0926127 and Beijing Municipal Natural Science Foundation 4122010 (2012.1 – 2012.12).
Authors' addresses: C. Xiao and Z. Huang, Computer Science Department, Beijing University of Technolory, Beijing, 100124, P. R. China; J. Cong, M. Gill, C. Liu, and G. Reinman, Computer Science Department, University of California, Los Angeles, West Los Angeles, Los Angeles, CA 90095; C.-C. F. Chang and H. Wu, Electrical Engineering Department, University of California, Los Angeles, West Los Angeles, Los Angeles, CA 90095. Correspondence email:xiaochunhua2011@gmail.com.

DOI 10.1145/2400682.2400719 http://doi.acm.org/10.1145/2400682.2400719

a power-efficient manner. However, efficient utilization of the on-chip communication bandwidth provided by these emerging interconnects still remains an open problem due to non-uniform and temporally irregular traffic patterns in current and future CMPs.

There are two main approaches to dealing with the allocation of the on-chip communication bandwidth for emerging interconnects. Both of them partition the aggregate bandwidth into a set of communication *channel*s. The first approach allocates these channels as application-specific shortcuts [Chang et al. 2008a] that are overlaid on the baseline traditional NoC to facilitate critical and heavy-loaded communication paths. This addresses the concern of spatial nonuniformity in NoC traffic. Offline compiler optimization or profiling methods are used to predict the communication pattern of program phases. Since the shortcuts are realized by tuning the frequency of the transmitter-receiver pairs, they can be reconfigured for each application or each phase of an application to match changes in program characteristics, and they can accommodate a degree of change in traffic patterns effectively. This is based on the assumption that the dynamic communication pattern changes can be predicted a priori, and are sufficiently rare to justify the cost of reconfiguring routing tables. However, in modern CMP designs, the data layout in the last- level cache (typically designed as NUCA, nonuniformed cache architecture) is dynamically determined by OS through virtual-to-physical page translation [Hardavellas et al. 2009]. Moreover, threads may be dynamically migrated to fully utilize the on-chip core resource [Constantinou et al. 2005] and cache blocks may also be dynamically migrated [Beckman et al. 2004] or replicated to reduce cache access latency. In a recently investigated composable accelerator-rich CMP [Cong et al. 2012 a], a virtual accelerator called by a thread is dynamically composed by the available on-chip accelerator building blocks, which cannot be predetermined. Moreover, in an accelerator-rich CMP, buffers may be dynamically allocated in any L2 cache bank based on the distance to the accelerator [Cong et al. 2012b]. All of these complications make it unrealistic to enable an accurate prediction of the on-chip communication pattern. Therefore, although this shortcut approach can efficiently address the spatial communication heterogeneity, it cannot effectively handle temporal communication heterogeneity.

The second approach uses token-based dynamic arbitration [Vantrease et al. 2008, 2009] to allocate the channels in real-time to communicating pairs on demand. Each receiver node in the NoC has its own channel which serves as its home node. The home node injects a token into its channel, and any senders that want to communicate to this home node can perform a destructive read of the token to acquire the communication channel, and then re-inject the token once it finishes this one-time communication. Unlike the aforementioned shortcut approach, this approach allocates the bandwidth on-demand at runtime with low arbitration latency, power, and hardware cost, satisfying the real-time communication requirement. If all of the receivers are uniformly receiving packets, it also tends toward high utilization and fair sharing. However, modern and future CMPs tend not to exhibit this uniformity due to the spatial communication heterogeneity.

Figure 1 depicts the high variation in number of received network flits per node for a NoC using RF-I with a token-based arbitration scheme. The results shown are complete executions of each benchmark (see Section 4.1 for a detailed system description). Each histogram denotes the received flits for one node over the total number of flits of the application (i.e., the percent of total flits received by a given node). The coefficient of variation for these applications ranges from 33% to 80%. This variation is even more significant if instantaneous demand is examined instead of aggregate demand.

In this situation, the channel bandwidth of the nodes that receive few packets is wasted, while the channels associated with the nodes receiving a large amount of packets are bandwidth-starved. It may be argued that the bandwidth allocated

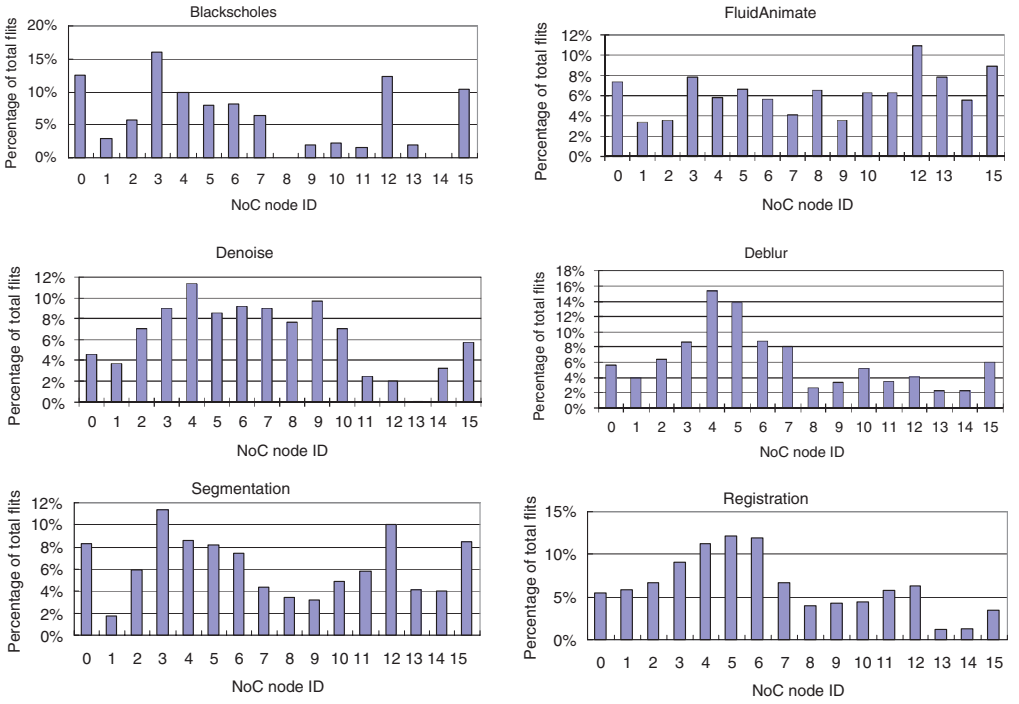

Fig. 1. Percent of received flits per node in a token-based RF-I NoC.

to each channel can be predetermined based on the home node's communication workload, but as mentioned earlier when discussing the shortcut approach, the temporal communication workload is difficult to predict a priori due to the dynamic uncertainties. Therefore, although this token-based dynamic arbitration scheme efficiently solves the temporal communication heterogeneity, it cannot effectively handle spatial communication heterogeneity.

Therefore, it is necessary to find an efficient bandwidth utilization scheme that can deal with both spatial and temporal communication heterogeneity. In this article, we propose a dynamic arbitration scheme called *Stream Arbitration*. Unlike token arbitration where channels are coupled to receivers, a channel in stream arbitration can be used to send packets from any sender to any receiver, which efficiently addresses the problem of spatial communication heterogeneity. Since stream arbitration is inherently a dynamic arbitration scheme, it also efficiently handles temporal communication heterogeneity. In this paper we choose RF-I to evaluate stream arbitration. RF-I is one promising alternative interconnect due to its compatibility with the existing CMOS design process, thermal insensitivity, low latency, and low energy consumption [Chang et al. 2008c]. But it should be noted that stream arbitration is not constrained to only RF-I: it can also be applied to other emerging interconnects, such as optical interconnects. The main contributions of this work can be summarized as follows.

—To the best of our knowledge, steam arbitration is the first dynamic arbitration scheme which targets emerging interconnect technologies that can efficiently deal with both spatial and temporal communication heterogeneity. Starvation avoidance and flow control are also carefully considered.
—We propose a detailed circuit-level design to realize stream arbitration in a radio frequency interconnect, with accurate modelling of area and power overhead.

—We develop a full-system cycle-accurate simulation infrastructure to evaluate the system performance and power impact of the stream arbitration scheme on various benchmarks and system configurations. We compare stream arbitration to token-based arbitration [Vantrease et al. 2009] (applied to RF-I), a representative bandwidth allocation scheme for emerging interconnects, and we show that stream arbitration can provide an average 20% performance improvement and 12% power reduction.

The remainder of this article is organized as follows: Section 2 describes the stream arbitration algorithm. Section 3 shows the physical feasibility of implementing a NoC that possesses the properties needed by stream arbitration on an example emerging interconnect technology: RF-I. Section 4 discusses the methodology we used to evaluate stream arbitration. We present experimental results in Section 5, and discuss the impact of stream arbitration on performance, power, and resource utilization. The scalability problem is addressed in Section 6, and related work is discussed in Section 7. Finally, we conclude our discussions in Section 8.

2. STREAM ARBITRATION: SCHEME AND EXAMPLE

In this section we describe stream arbitration from an algorithmic and architectural point of view. In Section 3, we will discuss the circuit support of the operations required by stream arbitration—such as stream augmentation and circulation in RF-I.

We partition the aggregate bandwidth provided by the RF-I or waveguide into several logical communication channels. One of them is used for arbitration; this is called the *arbitration channel*. The remaining channels are used for PE-memory data requests and responses; these are called *data channels*. Each RF node has one transmitter and receiver pair to access both the arbitration channel and the data channels. Active sources (nodes that want to send flits) compete for the data channels in the arbitration channel in order to talk to their desired destination nodes. Arbitration is done for each flit that is transmitted.

2.1. Stream Arbitration Scheme

The key component of our approach is the arbitration stream that travels across the arbitration channel. Conceptually, the arbitration stream starts at a single node, which is called the stream origin. The arbitration stream starts out logically empty and will travel in a unidirectional manner across all the nodes on the chip; this is called *Trip 1*. In this trip, when the stream passes each node, the node logically augments a number of bits (referred to as *substream*) in the arbitration stream to specify whether or not this node is attempting to send to another node, and whether or not this node is capable of receiving packets. It is important to note that these two pieces of information (desire to send and availability to receive) do not require any parsing of the stream—they only rely on information known a priori at the node. So there is no dependence where the stream must be read first and then modified. Such a dependence would impact arbitration latency by bringing slow logic on the critical path of the stream propagation.

In order to make sure the nodes only modify the stream without first reading it in the arbitration, each node has a specified region of bits that make up that node's substream, and substreams are disjoint within the arbitration stream. Collectively, these disjoint substreams will represent each node's interest in sending over a data channel and availability to receive from a data channel. The layout of a single substream element is shown in Figure 2. A node that wants to send a flit and is therefore contending for data channels is referred to as a *source node.* The destination ID is the label of the node to which a source node intends to send a flit (referred to as a *destination node*). The

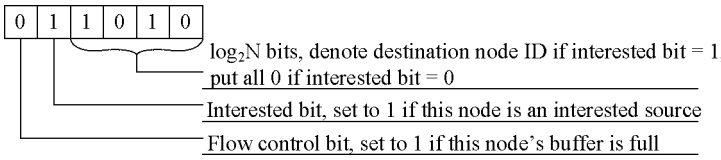

Fig. 2. The substream augmented by each node as the stream passed by.

flow control bit indicates whether or not there is sufficient buffer space at this node to accept a flit. N is the number of RF nodes.

After the arbitration stream passes the last RF node in Trip 1, it circulates over all nodes a second time, which we refer to as *Trip 2*. In this trip, when the stream passes each node, the node receives the arbitration stream but does not modify the stream. The purpose of Trip 2 is to parse the stream in order to check the following.

—*Ability to Send*. If this node is attempting to send a flit, information from the stream will be used to indicate whether this node can acquire a data channel, and if so, the data channel ID
—*Receive Channel*. Determine whether this node will be receiving a flit, and if so, the data channel ID where this data will be arriving is computed from the stream.

ALGORITHM 1: Stream Arbitration

Input: Stream: *flowControl*[1..N], *interested*[1..N], *destination*[1..N], where N is the number of RF nodes; the total number of channels M; this node's ID *node_id*.
Output: Transmitting_channel_ID, Receiving_channel_ID.

```
Transmitting_channel_ID = INVALID;
Receiving_channel_ID = INVALID;
channel_ID = 0;
for i = 1 .. N do
    if (interested[i] and (not flowControl[destination[i]])) then
        flowControl[destination[i]] = TRUE;
        channel_ID++;
        if (destination[i]== = node_id) then
            Receiving_channel_ID = channel_ID;
        end
        if (i = = node_id) then
            Transmitting_channel_ID = channel_ID;
        end
        if (channel_ID = = M-1) then
            break;
        end
    end
end
```

The algorithm is very simple and straightforward. It parses the stream in the order of the augmentation of the bits. A source node can acquire a data channel to a destination node if:

—the flow control bit of the desired destination node is zero,
—there is no upstream node already sending to the desired destination node. In this context, "upstream" means an earlier node in the unidirectional flow of the stream,
—there are still available data channels.

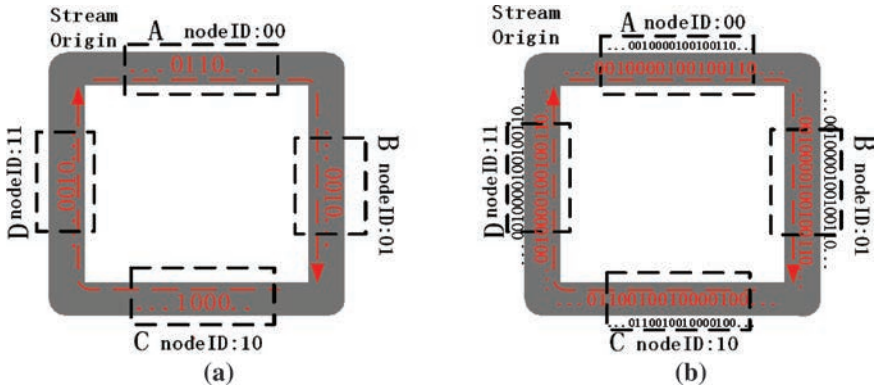

Fig. 3. An example of the stream arbitration scheme.

After the arbitration stream is parsed, the transmitter of a source node that has successfully acquired a data channel will be tuned to send on this channel, and the receiver of the intended destination node will be tuned to listen to the same data channel. A node can be a source and a destination simultaneously, using different channels. After Trip 2, the sources that successfully acquired data channels begin to use these data channels to communicate with their corresponding destinations. A channel is used for a single flit, and is surrendered after the flit transmitted. This does not incur a performance penalty because arbitration can be initiated every cycle, so a pair of nodes is allowed to communicate so long as the source continues to win arbitration. This requires pipelining stream arbitration and data transferring. The latency of the two-trip arbitration and the pipelining of the arbitration and data transferring in the physical design are detailed in Section 3.2.

One can see that the upstream nodes always have a higher priority in the arbitration than the downstream nodes (nodes encountered later in the unidirectional flow of the stream). In order to introduce fairness into stream arbitration, we use a rotating prioritization scheme, where each node is gradually reduced in priority each cycle, until it reaches the lowest priority. Each cycle, the lowest priority node from the previous cycle becomes the highest priority node. This prevents nodes that are lowest priority from being starved during periods of high system load. Moreover, each node gradually reduces priority to lessen the likelihood of burst data transfers dropping suddenly from highest priority in one arbitration cycle to lowest priority in the next. We found that this method allows for flits associated with multi-flit messages to arrive at the destination subsequently without high transmission latency deviation. From an architectural point of view, this gradual priority reduction is achieved by rotating the stream origin in the reverse direction of the stream traversal in the transmission line. However, we have a smart scheme detailed in Section 3.3 to support this without physically rotating the stream origin.

2.2. Example of Stream Arbitration

To illustrate our approach, this section presents a single example arbitration attempt (Figure 3). This example consists of only four nodes participating in arbitration, one arbitration channel, one data channel, and assume that the stream origin is at node A. Node A is attempting to send to node C, while nodes B and D are attempting to send to node A. C has no flits waiting for transmission, but has a fully occupied buffer and cannot receive any flits.

In Trip 1, each node augments its substream as described in Section 2.1 to the stream as showed in Figure 3(a) (the stream vectors follow the direction of red arrows in the figure). Nodes *A*, *B* and *D* are source nodes in this arbitration cycle. They modulate the interested bit "1" and their respective destination node IDs when the stream travels by. The node *C* has no requirement for data transmission, and its receiver buffer is unavailable for new messages in this arbitration. So node *C* only sets the flow control bit to "1" and leaves the interested bit at 0 when the stream travels by. Therefore, the substreams to be modulated by the nodes *A*, *B*, *C*, and *D* are "0110," "0100," "1000", and "0100," respectively.

In Trip 2, as shown in Figure 3(b), each node receives the full arbitration stream, and executes the algorithm presented in Section 2.2. Node *A* is the first node to modulate the stream and has the highest priority to acquire the data channel; but it cannot send because its destination, node *C*, has declared that it does not have available buffer space. Thus, node *A* loses in arbitration and does not receive a data channel. Then the next node in priority order, node *B*, wins the data channel in this arbitration since its destination, node *A*, has available buffer space. Node *C* does nothing, since its receiving buffer is full, and it is not an active node. From parsing the stream, Node *D* knows the upstream node *B* gets the data channel, and there are no more channels left. So node *D* also loses in this arbitration.

After the arbitration, the winner, node *B,* will transmit the message through the data channel, and others will retry in the next arbitration.

3. STREAM ARBITRATION IN RF-I

3.1. RF-Interconnect

RF-I was proposed in Chang et al. [2008b, 2008c] as a high-bandwidth, low-latency alternative to a traditional interconnect. Its benefits have been demonstrated for off-chip, on-board communication [Kim et al. 2012] as well as for on-chip interconnection networks [Wu et al. 2012]. The two most distinct advantages of RF-I compared to traditional interconnects are that: (1) instead of charging and discharging the whole wire to "1" or "0" as is done in a traditional electrical interconnect (which consumes substantial time and energy), RF-I modulates information on an electromagnetic carrier wave that is continuously sent along the transmission line; and (2) instead of trying to aggressively expand baseband bandwidth (which often involves power-hungry compensation technique to achieve a flat channel frequency response), RF-I divides bandwidth into frequency domains, each becoming a narrow-band signal which saves power. By doing this, RF-I also improves bandwidth efficiency by sending many simultaneous streams of data over a single transmission line. This particular technique is referred to as multi-band RF-I. Also, RF-I has been projected to scale better than traditional RC wires in terms of delay and power consumption and, unlike traditional wires, it can allow signal transmission across a $400mm^2$ die in 0.3ns via propagation at the effective speed of light. Figure 4 shows an exemplary RF-Interconnect link with ten bands transmitting on the same physical transmission line.

One key advantage of RF-I over traditional interconnects is its capability of multicast with on-chip directional couplers [Wu et al. 2012]. Impedance matched directional couplers eliminate the signal reflection that has inhibited multicast on traditional interconnects. Figure 5 shows an exemplary RF-Interconnect multicast link of one band. In the case that higher aggregate data rate is desired, more RF channels can be added in a similar fashion, as shown in Figure 4, with multi-band directional couplers. Since the required signal power scales with the number of drops along a multicast link, such as is shown in Figure 5, a larger amount of power is required to transmit a signal on a multicast link as compared to what is required for transmission on a point-to-point

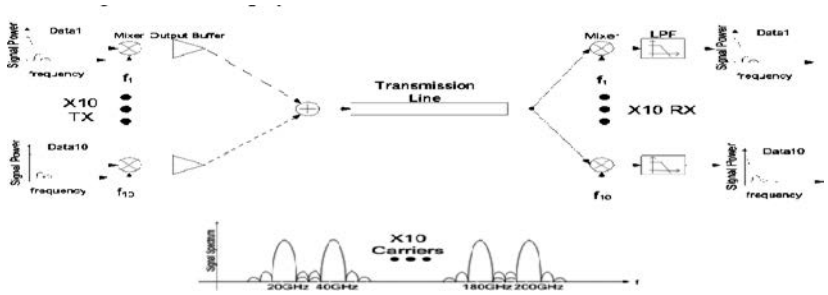

Fig. 4. A ten-carrier RF-Interconnect and corresponding waveform at the transmission line.

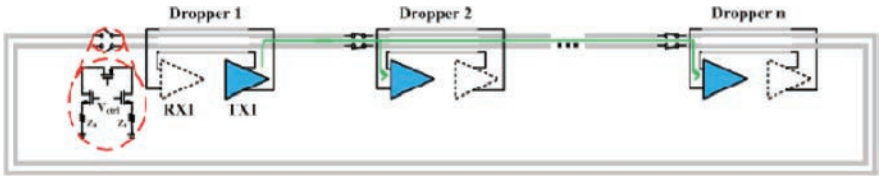

Fig. 5. Multicast RF-Interconnect system.

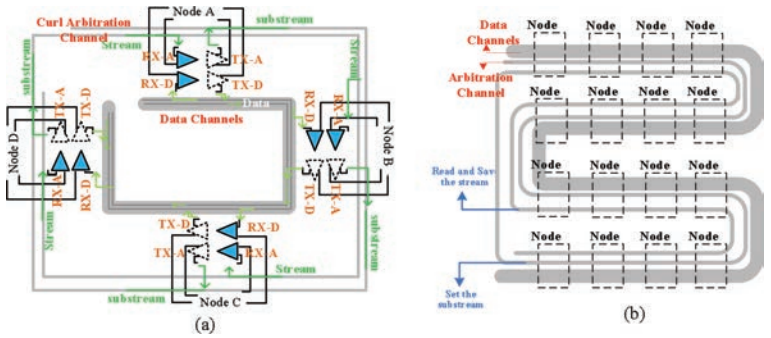

Fig. 6. The curl transmission line.

link [Tam et al. 2009; Wu et al. 2012]. Such effects are taken into consideration in our power estimation for transmission on the arbitration channel shown in Section 3.4.

3.2. Curl Transmission Line for Stream Circulation

A conventional RF-I transmission line is unidirectional and acyclic; that is, the starting point and the end point of the transmission line are two different points. This prohibits the stream circulation in the arbitration channel. To enable the two-trip stream arbitration, we propose a curled transmission line. Figure 6(a) shows the curled transmission line for the arbitration channel and the normal RF-I transmission line for the data channel. This curl starts from the stream origin at the outside loop and ends at the last RF node on the trip in the inner loop.

The outer loop of the arbitration channel is Trip 1, used for transmitting only, while the inner loop is Trip 2, used for receiving only. The transmitters to the arbitration channel (TX-A) of all the nodes are attached to the outer loop, while the receivers (RX-A) are attached to the inner loop. There is also a frequency-tunable transceiver pair (TX-D and RX-D) at each node, which is attached to the data channels. Although we presented a rectangular style transmission line in Figure 6(a), for better illustration

in the real physical design, all the transmission lines should go through each node, as shown in Figure 6(b). The reflection and discontinuity effect of sharp 90-degree turns in the curl transmission lines can be mitigated by careful designs for impedance matching. For example, Wu et al. [2012] used 45^0 diagonal routing at each corner of the channel to eliminate sharp turns. As a result, this did not impact the interconnect performance. Depending on different CMOS fabrication technology, rounded turns can also be implemented to better avoid the reflection issue caused by sharp turns. In our evaluated system, which is a $1cm^2$ chip with 16 PE clusters (each cluster has one RF node), the total distance for one trip in the arbitration channel, or the longest distance in the data channel, is 5cm. The speed of light in silicon is 8ps/mm; thus each trip of the arbitration can be finished in 400ps. Our evaluated system has a working frequency of 2GHz. Therefore, each trip only takes one cycle, and any flit transfers on the data channel can reach their destination in one cycle.

During any particular cycle, the TX-As of the nodes are augmenting their substreams for the arbitration initiated during cycle x; the RX-As of the nodes are receiving the entire stream for the arbitration initiated during cycle x-1; the local stream parsing unit is parsing stream for the arbitration initiated during cycle x-2; the RX-Ds and TX-Ds of the winning sources of the arbitration initiated during cycle x-3 are using data channels to transfer data. In this way, stream arbitration can be initiated every cycle.

3.3. Time Division Modulation Multicast For Stream Augmentation

We propose a time division modulation multicast (TDMM) approach in the arbitration channel to achieve stream augmentation with priority rotation. The latency T for one stream trip can be divided into N slots, where N is the number of RF nodes, and the length of each slot is $\lambda = T/N$. Let $d(v)$ denote the RF node hops from the stream origin to node v. Let $p(v)$ denote the priority of node n in a particular arbitration, in which "p = 0' means highest priority. Then, the slot for a node to modulate its substream in the arbitration channel is $(p+d)\ \lambda$.

An example of TDMM is shown in Figure 7. In this example, there are four RF nodes. Node A is the stream origin where the curl starts. Assume the current highest priority is rotated to node C, and the priority order is in the reverse of the stream travel direction. Then we have: node A: $d = 0$, $p = 2$; node B: $d = 1$, $p = 1$; node C: $d = 2$, $p = 0$; node D: $d = 3$, $p = 3$. Therefore, nodes A, B, C, and D will modulate their substreams at slots 2λ, 2λ, 2λ, and 6λ, respectively. These modulated substreams will finally form a stream that is in the order of the priority of their nodes when the stream makes the second pass to be read.

The proposed TDMM approach can support any arbitrary priority assigned to these nodes using the formula described previously, provided all nodes have unique priorities. Here, we adopt the gradual priority reduction scheme discussed in Section 2.1. Each node locally keeps a small counter to record its current priority p. Initially, each node v is assigned with a priority $p(v) = d(v)$. After each arbitration, it increases its $p(v)$ by 1. When $p(v)$ reaches N, where N is the number of RF nodes, indicating that it is at the lowest priority, the node will reset $p(v)$ to 0, which is the highest priority in the next arbitration.

In Trip 2, each node also receives one substream per slot. A node does not wait for all the substreams to begin parsing the stream (using the algorithm in Section 2.1). Instead, each time a node receives a substream, it begins to process the substream by performing one iteration in the algorithm. The only difference is that the flow-control bits are buffered but not checked, and are used at the end to invalidate the winning source nodes with destinations that have a full buffer. In this way, we parallelize the stream receiving and stream parsing.

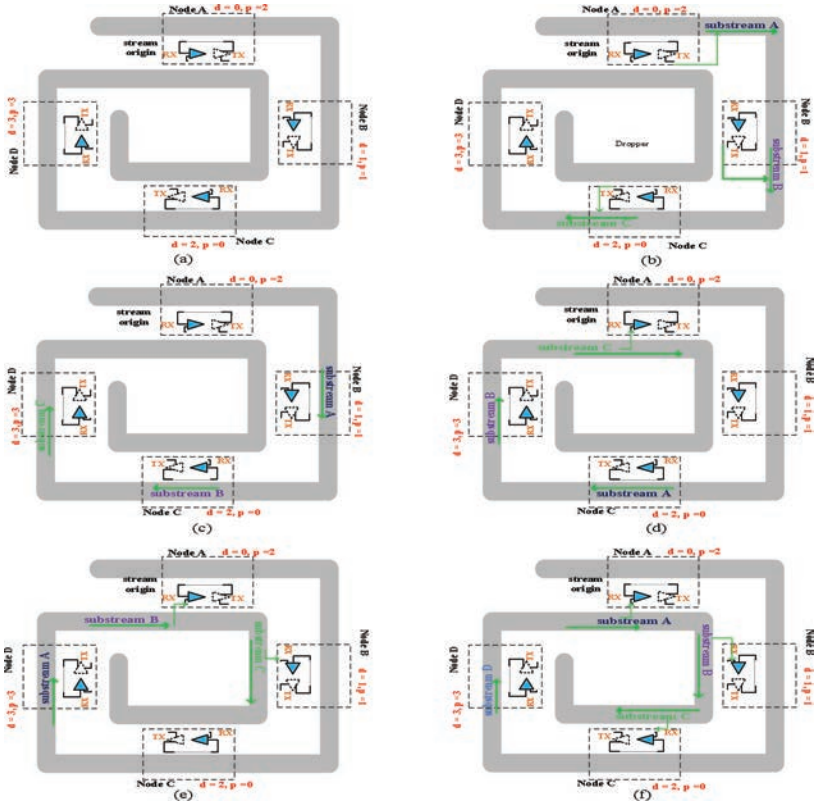

Fig. 7. An example of time division modulation multicast for stream augmentation with priority rotation: (a) $t = 0$ and $t = \lambda$: no substream is modulated. (b) t $= 2\lambda$: nodes C, B, A modulate their substreams simultaneously. (c) $t = 3\lambda$: substream C, B, A achieves nodes D, C, B, respectively. (d) $t = 4\lambda$: substream C, B, A achieves nodes A (inner loop), D, C, respectively. Substream C is received by node A. (e) $t = 5\lambda$: substream C, B, A achieves nodes B (inner loop), A (inner loop), D, respectively. Substream C and B is received by nodes B and A, respectively. (f) $t = 6\lambda$: node D modulates its substream. Substream C, B, A achieves nodes C (inner loop), B (inner loop), A (inner loop), and is received by them, respectively.

3.4. Power and Area

For power estimation, the predicted power parameters are different between the arbitration channel and data channels. Although data channels are broadcast links, the arbitration strategy allows them to be treated as point-to-point links in any data communication cycle. On the other hand, due to large signal attenuation for the multicast link in the arbitration channel, increased power is needed to meet the signal-noise-ratio (SNR) requirement of the desired bit-error-rate (BER, 10^{-12}).

The power and area modeling values of point-to-point RF transceivers used in this article and implemented with 32nm CMOS technology are shown in Table I. The TX and RX power consumption values are predicted (scaled) from our implementation of a multi-band RF-I at 90nm CMOS technology [Wu. et al. 2012]. For the scaling from 90 nm to 32 nm CMOS process performance, it is assumed that the average power consumption per transceiver channel is expected to stay constant at about 6mW. The logic behind the assumption is that although RF circuits at higher carrier frequencies require more power, this additional power is compensated by the power saved at the lower carrier frequencies due to higher f_T transistors available with scaling. In addition to the increased number of channels, the modulation speed of each carrier would also

Table I. Power Parameters of Point-to-Point RF Transceiver in 32nm Technology

	Power (mW)	Power Efficiency (pJ/b)	Active Area	Passive Area
TX Mixer	1		5um × 5um	0
TX PA	2.5		10um × 10um	50um × 50um
Total TX	3.5	0.44	$125um^2$	$2500um^2$
RX Mixer	0.5		10um × 10um	50um × 50um
RX Baseband	2		20um × 20um	0
Total RX	2.5	0.31	$500um^2$	$2500um^2$

Table II. Power Parameters of Arbitration RF Transceiver in 32nm Technology

	Power (mW)	Power Efficiency (pJ/b)	Active Area	Passive Area
TX Mixer	1		5um × 5um	0
TX PA	5		15um × 15um	50um × 50um
Total TX	6	0.6	$250um^2$	$2500um^2$
RX Mixer	3.5		20um × 20um	50um × 50um
RX Baseband	2		20um × 20um	0
Total RX	5.5	0.55	$800um^2$	$2500um^2$

increase, allowing a higher data rate per channel. As a result, the data rate per channel per wire is predicted to be 8Gbps, which results in a power efficiency of 0.75pJ/b. A behavioral model simulation shows that 15GHz channel spacing is sufficient to carry 8Gb/s data with a low BER. Therefore, it is projected that 12 carriers can be sent simultaneously on each transmission line given the 350GHz f_T of 32nm technology; this indicates a 96Gbps aggregate data rate on each wire. The active area and passive area are also predicted from our 90nm multi-band RF-I prototype.

Table II shows our power and area modeling of arbitration RF transceivers in 32nm CMOS technology. The power consumption is estimated by scaling our implementation of a multicast RF-I at 65nm CMOS technology in a fashion similar to the scaling of point-to-point RF transceivers. The power efficiency is predicted to be 1.15pJ/b. The number is higher than that of point-to-point RF transceivers mainly because of the larger channel loss of multicast data links. The data rate per channel per wire is predicted to be 8Gbps. Therefore, 12 carriers provide an aggregate bandwidth of 96Gbps on each wire for the arbitration channels. The active devices area is also less than 2x larger than the point-to-point link because of the higher gain required for arbitration links (larger devices implemented).

Due to the power overhead being basically the charging and discharging of the bias transistor's gate capacitance, we configure the RF transmitter and receivers with simple logic gates that control their bias stage, so the RF transmitter and receivers can be turned off to save power when they are not in use. For example, a 50fF gate capacitance of the receiver bias transistor indicates a 25fF energy consumption when turning it on and off, which is substantially smaller than demodulating one bit from the arbitration channel (>1pJ/b). The speed of this power switching depends on the driving strength of the controlling logic gates. In 32nm technology, this speed is expected to be well below 0.05ns.

4. EVALUATION METHODOLOGY

4.1. Simulation Infrastructure

We evaluated stream arbitration using a composable heterogeneous accelerator-rich multiprocessor (CHARM) [Cong et al. 2012], where the on-chip accelerator building blocks (ABBs) can be dynamically composed into virtual accelerators based on application requirements. While this architecture only features a small number of cores, the

Table III. Evaluated System Configuration

Parameter	Value
Processor	8 2.0GHz Ultra-SPARC-II-I
Operating System	Solaris 10
Private L1 Cache	32-KB 4-way set-associative, 1-cycle access latency for each core
Shared L2 Cache	4-MB 8-way set-associative, 10-cycle access latency in 64 banks
Coherence protocol	Shared banked uniformly distributed MOSI L2-Cache, private MSI L1-cache for cores, Distributed directory.
Memory	4 Memory controllers, 450 cycle latency, 30GBPS bandwidth each [120 GBPS aggregate]

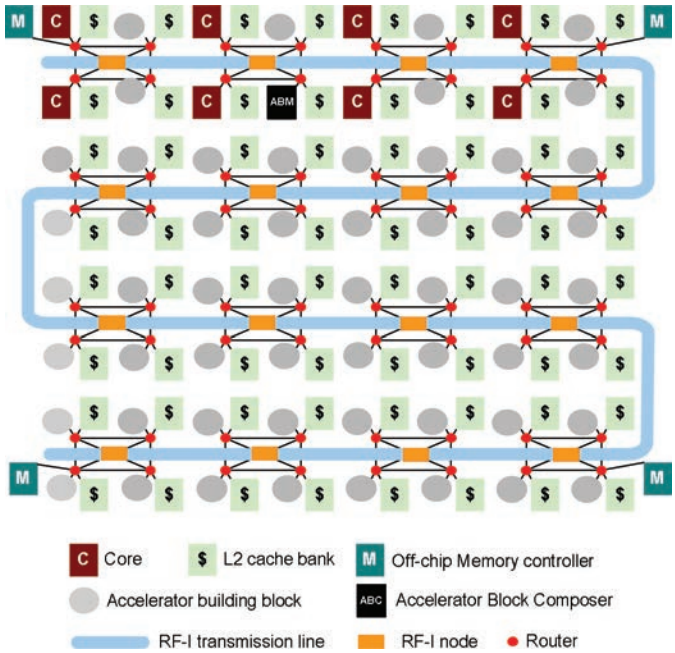

Fig. 8. Overall diagram of the evaluated CHARM architecture with RFI overlaid NoC.

network and memory demands of the various accelerator components are substantially larger than that of a system featuring only conventional cores. We chose this architecture because we recognized that the NoC design will not advance in isolation, and in order to evaluate a future NoC, we should evaluate it in combination with a system that will place a demand on the NoC that is representative of a future architecture targeting HPC-style applications, which are characterized by high throughput with low power consumption. As shown in Figure 8, our modeled system consisted of multiple cores, an accelerator block composer (ABC), a set of ABBs, memory controllers, and L2 cache bank nodes. These nodes are arranged, in all configurations, in an 8-by-8 grid, resulting in 64 nodes, with accelerators uniformly distributed. Table III provides details of the system configuration. These details are consistent across all configurations.

We have extended the Simics [Magnusson et al. 2002] and GEMS [Martin et al. 2005] simulation platforms to model an RF-I network, along with the stream arbitration scheme. As a point of comparison, we have also implemented a recently proposed token-based arbitration scheme for performing arbitration over optical interconnects [Vantrease et al. 2009], which we will refer to as *token arbitration*. Our implementation

of token arbitration was adapted to use a RF-I network so as to make a fair comparison. Brief details of the operations of token arbitration are described in Section 4.3. Timing, area, and power consumption associated with the RF-I network was measured using the methodology discussed in Section 3.4, and was incorporated into our cycle-accurate simulation platform. Timing, area, and power consumption associated with the digital circuitry required to perform arbitration was obtained by compiling code implementing our arbitration scheme into RTL using the AutoPilot [Cong et al. 2010] behavioral synthesis tool, which was then synthesized using the Synopsys Design Compiler. For network communication along traditional wires, such as that between multiple nodes and shared RF-I transmission points, we used Orion 2.0 [Kahng et al. 2009] to estimate power consumption in 32nm technology. Accelerators and accelerator arbitration mechanisms used in this work were modeled after those presented in Cong et al. [2012]. Accelerators were included due both to the expected increase in their use in future processors, and as an effective way of stressing the NoC in a way that cores cannot. The accelerator design methodology and accelerator arbitration mechanism we adopted for this work supports hardware load-balancing.

To more effectively examine the performance of stream arbitration and how it performs relative to token arbitration, we examined a number of different system configurations and RF-I configurations. We modeled cases in which nodes on our network were bunched into clusters of varying sizes, with RF-I serving as the only communication mechanism between clusters. We also scaled the capability of our examined RF-I network in terms of number of channels and bandwidth per channel. A clustered architecture means several routers share one RF transceiver. Only the communications between different clusters will go through the RF-I, with traffic between nodes in the same cluster instead using a fully connected traditional network. We considered the impact on latency and power of different cluster sizes of 4, 2 and 1, and found that the effect is minor due to marginal reduction in the amount of bandwidth dedicated to servicing data channels when reducing the cluster size. Another reason is that, from the perspective of each individual cluster, the overwhelming majority of network traffic was inter-cluster traffic, irrespective of the number of nodes in each cluster. Therefore, we chose a cluster size of 4 as our baseline configuration in this article if not otherwise specified.

4.2. Benchmarks

To evaluate our work, we examined a number of accelerators using benchmarks. These benchmarks consisted of those evaluated in the CHARM design [Cong et al. 2012], along with two PARSEC benchmarks [Bienia et al. 2008], Fluid Animate and Black Scholes, that were amenable to being mapped to accelerators. The examined region was restricted to the program kernel, which was either entirely, or nearly entirely, covered by hardware acceleration.

Since our selected accelerator arbitration scheme features hardware load-balancing among accelerators, there is not a substantive difference in system load when adding software threads. For this reason, our benchmarks were primarily executed sequentially, with the accelerated regions relying on load-balancing hardware to achieve concurrency.

4.3. Reference Scheme

Token arbitration, presented in Vantrease et al. [2009], is a multiple-writer single-reader arbitration mechanism designed for use in optical NoCs. Token arbitration makes an effort toward efficiently solving the problem of arbitrating a shared communication channel to ensure that multiple writers on a single channel do not interfere with one another. In order to ensure that only a single writer is writing to a given

destination, the destination node transmits a send token around a ring in the NoC. A node interested in sending will halt the progress of this token by reading it, and hold it until it finishes sending, at which point it will re-emit the token. This requires that there is a single communication channel for each potential destination node.

While token arbitration makes guarantees of liveness and fairness, restricting each communication channel to service only a single destination potentially results in poor channel utilization and difficulty scaling with a number of nodes, since the number of channels must also scale accordingly. We chose token arbitration as a point of comparison because it is recent work, and it targets the specific problem of sharing communication channels on emerging interconnect technologies.

While static shortcuts [Chang et al. 2008a], as mentioned in Section 1, are also designed to utilize emerging interconnect technologies, we chose not to compare against this design point due to the fundamentally different nature of the problem that static shortcuts address. Static shortcuts also rely on an underlying mesh network as a fallback option if a shortcut isn't present, while our approach does not. As a result, it would not be possible to construct a fair comparison between static shortcuts and stream arbitration.

5. RESULTS AND DISCUSSIONS

5.1. Performance as Bandwidth Scales

Figure 9 shows a comparison of average network flit latency between stream arbitration and token arbitration for various network configurations. Each system consists of 16 clusters of 4 nodes each, laid out in a grid, with a single RF-I access point for each cluster. Aggregate bandwidth is calculated by the number of channels multiplied by the bandwidth of each channel. Token arbitration requires a single channel for each possible destination, and thus aggregate bandwidth is adjusted by scaling the bandwidth of each channel from 2 to 16 bytes per cycle. Stream arbitration decouples the number of nodes from the number of channels, so instead, aggregate bandwidth is adjusted by fixing the bandwidth of a single channel at 16 bytes per cycle and scaling the number of channels from 2 to 16. As expected, the performance of both systems becomes near equal as both the bandwidth per channel and number of channels becomes 16 for both arbitration schemes. Our experiments also clearly show this.

As is shown, there is an advantage to having a few channels of high bandwidth, and allowing them to be flexibly used by any communication pairs at runtime, over having many channels of uniformly low bandwidth. This is intuitively sound, because for each cycle only a fraction of nodes will be communicating with one another. Having a large number of channels is only an advantage when the traffic pattern is highly uniform, at which point all channels can be utilized continually. Since each destination can only have a single channel associated with it at a time, any nonuniformity in the traffic pattern, even if just instantaneous, results in favoring a few high bandwidth channels. While both systems were able to reduce average flit transmission latency by upwards of 60–70% for most benchmarks in systems that feature an abundance of RF-I resources, we found that stream arbitration was able to approach this figure much more quickly when RF-I resources were reduced.

For most of our examined benchmarks, the benefit of adding channels diminished quickly, highlighting that only a small handful of nodes at any given instant are responsible for the majority of the NoC traffic. While cache activity was approximately balanced, clusters in the corners of the network which featured memory controllers saw additional traffic in the form of memory requests, and clusters featuring highly utilized accelerators saw additional traffic in the form of cache responses.

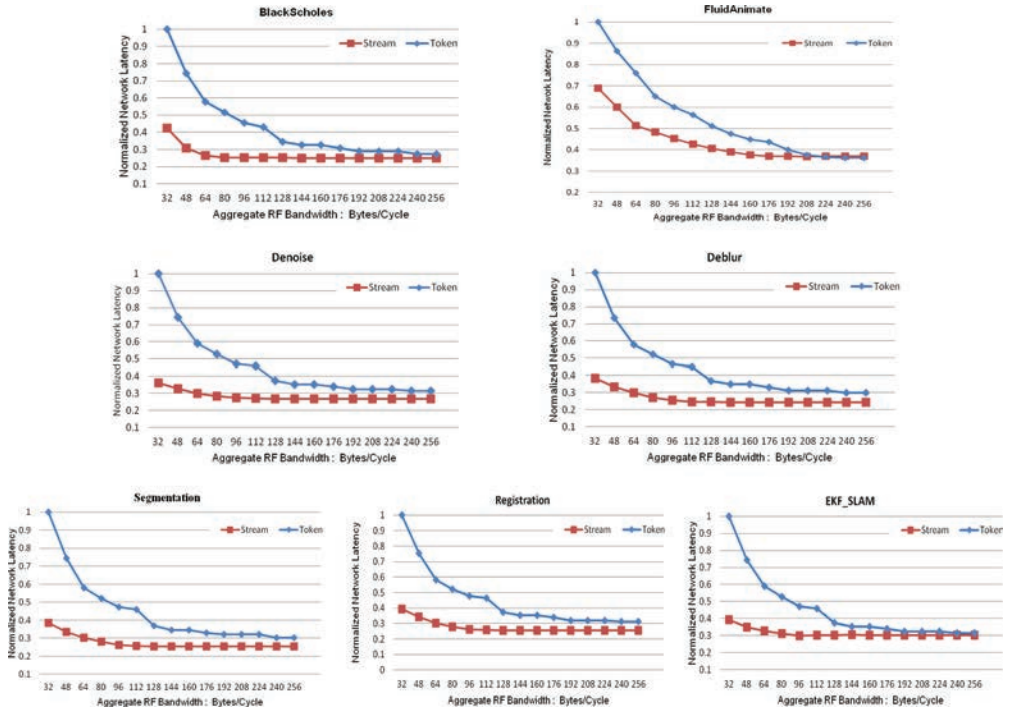

Fig. 9. Comparison results of average network flit latency at various aggregate bandwidths (normalized to token arbitration with 32 byte/cycle aggregate bandwidth).

Figure 10 shows the impact of this latency reduction on benchmark runtime. NoC latency reduction doesn't correlate directly with performance primarily due to all of the other factors contributing to performance that are not NoC related. Chiefly among these, for both accelerator- centric architectures examined in this work and conventional CMP designs, is the contribution of memory latency. Reducing NoC latency primarily contributes to performance by reducing the latency associated with coherence between the private and shared cache layers. While there is a correlation between a reduction of NoC latency and improvement of overall performance, the 60–70% NoC latency reduction we observed contributed only 15–25% runtime reduction in most cases. Some benchmarks were outliers, such as Deblur, which is a memory-bound benchmark that features a working set that fits in shared cache. In these cases, we observed a much more pronounced relationship between NoC flit latency reduction and performance improvement.

5.2. Energy Consumption

Energy in emerging NoC designs, such as RF-I, comes primarily from two sources. First is modulation and demodulation of the signal, which occurs when a "one" bit of data is sent on the network or received from the network. Second is the preparation of the communication medium for carrying data. For RF-I this involves supplying receivers with energy to listen to the attached channel, which must be done perpetually for any channel on which data may arrive, whether the given channel is being used to carry data or not. These two sources are analogous to dynamic and static power on traditional electric circuits. Results shown in Figure 11 were gained from the experiments for

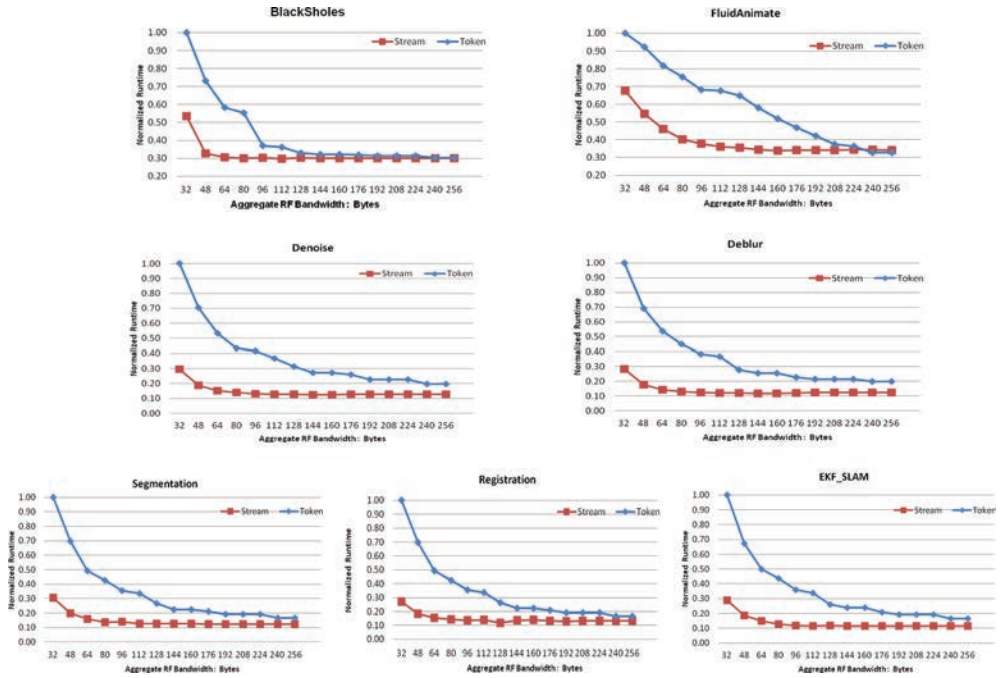

Fig. 10. Comparison results of application runtime at various aggregate bandwidths (normalized to token arbitration with 32 byte/cycle aggregate bandwidth).

which performance results were shown in Section 5.1. Energy consumption for CPUs, caches, and accelerators is not shown in order to highlight the contrast between the two arbitration algorithms.

The RF transmitter and receivers can be turned off to save power when there is no RF signal modulation and demodulation, as discussed in Section 3.4. Therefore, there is power savings when there is no data being sent to/from a RX-D/TX-D through the data channels, or there is no arbitration in the arbitration channel. Token arbitration, on the other hand, continually circulates the arbitration tokens, even when the NoC is otherwise unused. The power required to do this scales linearly with the number of potential destination nodes, but is generally small relative to the power required to actually send data. A considerable portion of the energy difference is attributed to the difference in the arbitration success rate. Stream arbitration features a very high arbitration success rate in general due to the short utilization period of a given channel. In comparison, token has a relatively poor arbitration success rate due to sending nodes holding the token for long durations when link bandwidth is small.

As shown in Figure 11, the difference in energy consumption between stream arbitration and token arbitration is mostly related to the difference in performance provided at a given design point. But token performs better than stream at points that provide similar or identical performance. The reason for this is that stream has more modulation power than token, which needs to modulate 6 bits each time compared to only 1 bit for token.

5.3. Discussion 1: Bandwidth Allocation for Channels

To further examine the point discussed in the previous section, we conducted an experiment with a fixed aggregate bandwidth, and adjusted the number of channels and

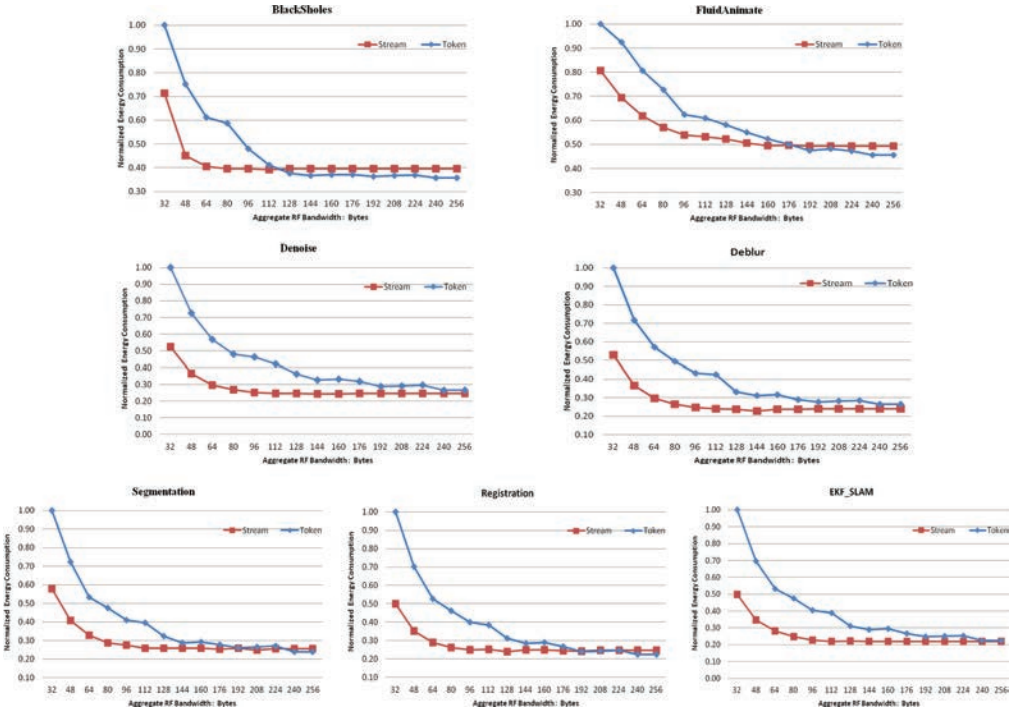

Fig. 11. Comparison results of application network power consumption at various aggregate bandwidths (normalized to token arbitration with 32 byte/cycle aggregate bandwidth).

channel bandwidth to achieve that aggregate bandwidth. Figure 12(a) shows the impact on the average flit latency of adjusting the division of bandwidth into multiple channels, with a fixed aggregate RF-I bandwidth budget of 108 bytes per cycle. The optimal configuration varies by benchmark. While Section 5.1 showed that adding additional bandwidth was never observed as resulting in a degradation of performance, it is clear that there is a compromise to be found between high bandwidth channels and additional channels. Figure 12(b) shows the corresponding impact on execution time for each design point.

While, in this work, we focused on static partitioning of bandwidth into multiple channels, this figure shows the potential for dynamically partitioning this bandwidth as well.

5.4. Discussion 2: Data Channel Utilization

Figure 13 shows the percent of total RF-I traffic that occupies each data channel using stream arbitration for several selected benchmarks, for a system with 64 nodes and clusters of size 2, which is allocated 6 RF data channels and 16 bytes per cycle bandwidth per channel. As described in Section 2, stream arbitration allocates data channels to a communicating source-destination pair ordered by winning arbitration. Thus, if any data at all is being sent over RF-I, it is guaranteed that data channel 0 is in use, followed by channel 1 if two channels are in use, and so on. This figure shows that even a single channel can accommodate between 27% and 52% of the total bandwidth demand. The utilization of channels drops steadily. While data could be shown for systems with larger numbers of channels, the utilization for later channels becomes

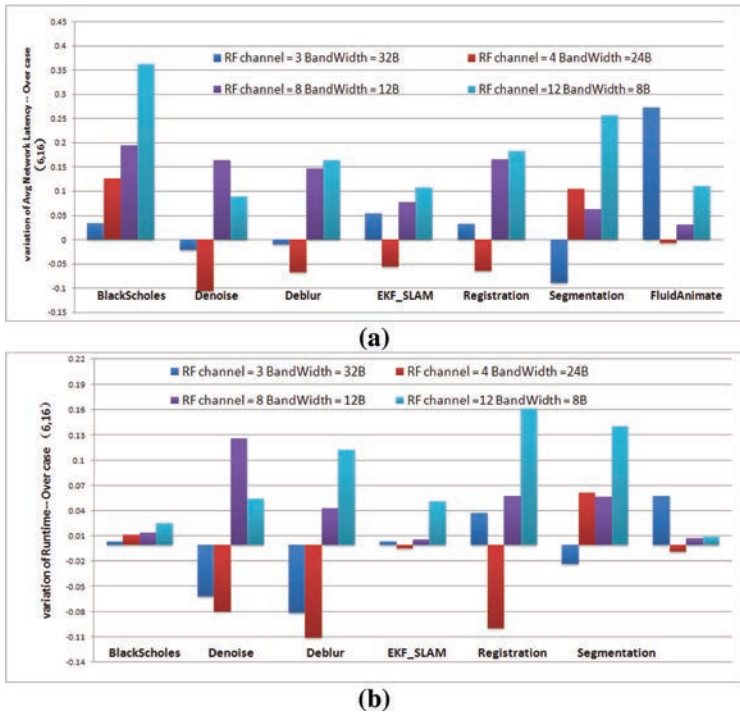

Fig. 12. Impact of channel bandwidth allocation: (a) Average network flit latency. (b) Application runtime (results are the variation over the case of "RF channel = 6, Bandwidth = 16 byte/cycle").

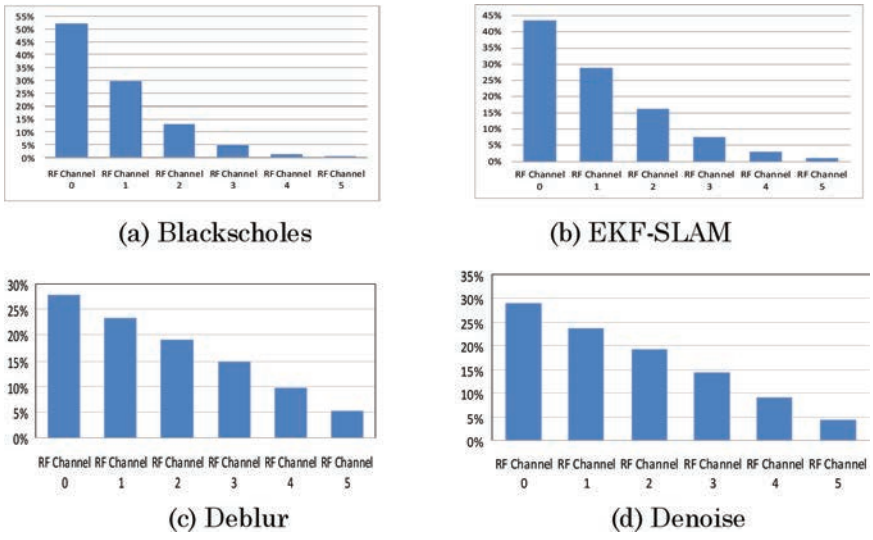

Fig. 13. RFI data channel utilization.

extremely low, indicating that the physical RF-I medium is being wasted. This clearly indicates that dynamic channel allocation is critical to effective resource utilization of emerging networks, such as RF-I, as static allocation results in a large amount of waste.

6. SCALABILITY

6.1. Hierarchical Stream Arbitration

As the chip size scales up or the number of RF nodes increases, the length of the transmission line increases, and completing a single stream trip may need P cycles, where P is larger than 1. In this case, the latency to complete arbitration would increase to $2P+1$. As the number of RF nodes N increases, the amount of data sent to perform a single arbitration increases in $O(N\log N)$. If we keep using a single curl to perform flat stream arbitration for all RF nodes, we will encounter serious challenges in terms of performance, power, and area.

Performance. Each arbitration needs $2P+1$ cycles to finish; therefore, it will take a much longer time for a network message to claim a data channel successfully as P becomes large. This results in an increase in average network latency. While we can still use pipelining to initiate stream arbitration in every cycle in order to guarantee throughput, average network latency can have a significant impact on system performance for workloads that feature message dependencies. Under this circumstance, the long $2P+1$ arbitration latency limits the scalability of the base stream design.

Power. The length of the arbitration stream increases in $O(N\log N)$, and the power requirement of the arbitration channel also increases in $O(N\log N)$. If N is a relatively small number, then compared to the data transfer power in the data channels, the arbitration power is negligible. This is how stream arbitration can win over token arbitration in terms of power. However, as N becomes large, then this arbitration overhead becomes nontrivial compared to data transfer power.

Area. As the number of data channels increases as N scales up to maintain network performance, the total number of channels that should be supported by each RF transceiver connected to data channels also increases linearly with N. For this reason, the area of the communication substrate increases quadratically with N.

In large-scale CMPs, it is no longer the case that all the network elements are communicating with all other elements uniformly. The state-of-the-art nonuniform cache architecture (NUCA) management schemes, such as RNUCA [Hardavellas et al. 2009] and page-recoloring scheme [Cho and Jin 2006], intend to place cache blocks near their most frequent requestors by smart initial placement, dynamic migration, and replication. Moreover, cache partitioning schemes [Qureshi and Patt 2006; Lee et al. 2010; 2011] can be used so that a cluster of cores are only going to access a locally allocated cache partition. Communication between these partitions is only required when there are coherence invalidations and fills. Therefore, it is expected that in future large-scale CMPs, the local communication in a core-cluster will dominate the overall communication. Under this circumstance, a flat stream arbitration scheme that uses a single curl to go across all nodes is wasting resources and unnecessarily limiting performance. Here, we propose a hierarchical stream arbitration scheme to make use of the communication patterns expected to be found in future large-scale CMPs. In the hierarchical stream we have a local transmission line (TL) for each core-cluster, and a global TL to connect these local TLs. Each TL consists of a curled arbitration channel and a set of data channels, as shown in Figure 14. The stream arbitration is performed independently in each level of the hierarchy.

The local TL is connected to the global TL through a relay node that consists of two RF interfaces and two buffers, as shown in Figure 14. Each RF interface is similar to the RF nodes described in Section 3.2. One RF interface consists of a pair of transceivers (TX-A and RX-A) connected to the local curl and a pair of transceivers (TX-D and RX-D) connected to local data channels. Another RF interface also has two sets of transceivers connected to the global curl and global data channels. Buffers are used to temporarily buffer the network messages which have already arrived at the relay node but are still

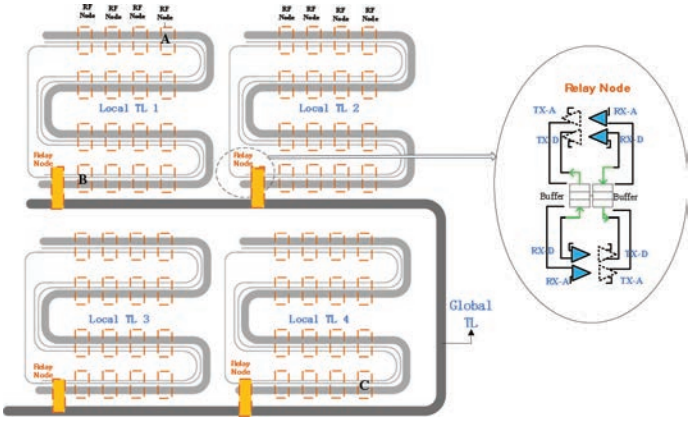

Fig. 14. Hierarchical stream arbitration.

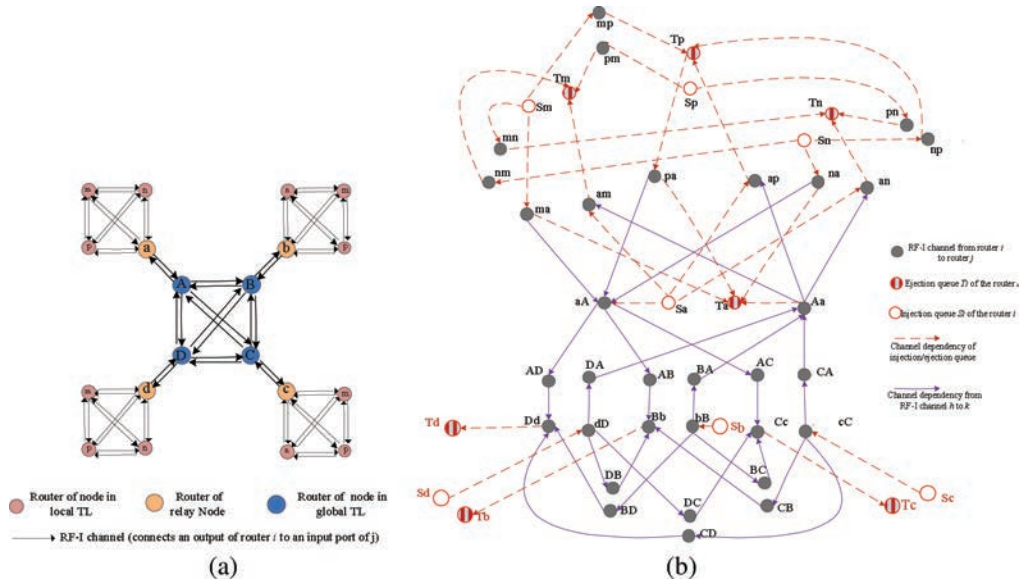

Fig. 15. Hierarchical NoC topology characterization graph of (a) *TG*, (b) *CDG*.

waiting for arbitration success to move forward—either from the local TL to the global TL, or vice-versa. Each of the buffers will attach its flow control signal to the substream in the corresponding arbitration channel in order to indicate the fullness of this buffer, as a normal RF node does in Section 2.1.

The proposed hierarchical stream arbitration RF-I NoC is deadlock-free as long as we only allow minimal routing. Figure 15 shows the topology graph (TG) and the corresponding channel dependency graph (CDG) of an example hierarchical stream arbitration RF-I NoC (the definitions of both TG and CDG are detailed in Cong et al. [2010]). In this example there are four local transmission lines (TL) and one global TL. Each local TL contains four RF nodes (one of them is the relay node). In the TG, when we say that there is a channel (the unidirectional arrows in Figure 15(a)) from RF node A to RF node B, we mean that a packet can go directly from RF node A to RF node B. Due to space limitations in the CDG (Figure 15(b)), we only detail the channel dependencies

in the global TL and one of the four local TLs. This is sufficient because the CDGs of the four local TL are identical. The purple solid arrows in Figure 15(b) denote the channel dependencies among the channels in TG, and the red dashed arrows denote the channel dependencies between the channels in the TG and the network injection queue (Si) and ejection queue (Ti). As can be seen in Figure 15(b), the CDG of the network is acyclic. According to the theory in Duato et al. [2001], the routing of the proposed hierarchical stream arbitration RF-I NoC is deadlock-free.

The example shown in Figure 14 has 8x8 RF nodes (actually 256 network nodes and each four network nodes share one RF transceiver). We let 16 RF nodes share one local TL, so that we can have enough slack to enable one stream trip to finish in one cycle, as discussed in Section 3.2. The total distance for one stream trip in global TL is 5cm with this 8x8 RF node topology, and so it can be finished in one cycle also in this topology. This two-level hierarchical design contains four local TLs and one global TL. Since there is also a relay node in the local TL, there are actually 17 RF nodes competing for the data channel resources of the local TL. In the global TL, there are four RF nodes (the four relay) competing for the data channels resources of global TL. In the example shown in Figure 14, if node A wants to send a message to node B, it only needs to perform arbitration within its local TL, where each arbitration takes three cycles (one for the first stream trip, one for the second stream trip, and one for the stream parsing). Then the message can be sent to B using the granted data channel from this local arbitration. However, if node A wants to send a message to node C, it first attempts local stream arbitration in Local TL 1, and then uses the granted local data channel to send the message to Relay Node 1. In Relay Node 1, the message is buffered and waiting for the global arbitration to allocate it a data channel in Global TL. Upon success, the message is sent by Relay Node 1 to Relay Node 3 using the granted global data channel. Then in Relay Node 3, the message is buffered again and waits for the success of arbitration in Local TL 3, at which point it is sent to node C by Relay Node 3. The entire process takes at least nine cycles.

6.2. Trace-Driven Evaluation Methodology

To evaluate the performance of the hierarchical architecture, we scaled the NoC size (in terms of number of routers) to 16x16, 24x24 and 32x32. In these topologies, each 2x2 routers share one RF node. The three evaluated NoC designs have 8x8, 12x12, and 16x16 RF nodes, respectively.

—For hierarchical stream arbitration, every 4x4 RF nodes share one local TL, so that in each local TL there are 17 RF nodes in total (the additional one node is the relay node). One trip in the curl of the local TL takes one cycle. The hierarchical architecture of 8x8 RF nodes is already shown in Figure 14, where the global TL has four RF relay nodes. The hierarchical architecture of 12x12 and 16x16 RF nodes is shown in Figure 16, where the number of RF relay nodes in their global TLs are 9 and 16, respectively. One stream trip on the curl of the global TL for these three designs takes 1, 2, and 4 cycles, respectively (depends on the length of TL).
—For flat stream arbitration, there will be a single TL across all of the RF nodes for each NoC design. With the assumption and discussion in Section 3.2, one stream trip in the curl of the single TL for the three NoC designs takes 3, 7, and 11 cycles, respectively (depends on the length of the TL).

We assign two 16B data channels for each local TL for the hierarchical stream arbitration. To make a fair comparison, we make the number of data channels in the TL of flat stream arbitration the same as the total number of local TL data channels of the corresponding hierarchical design.

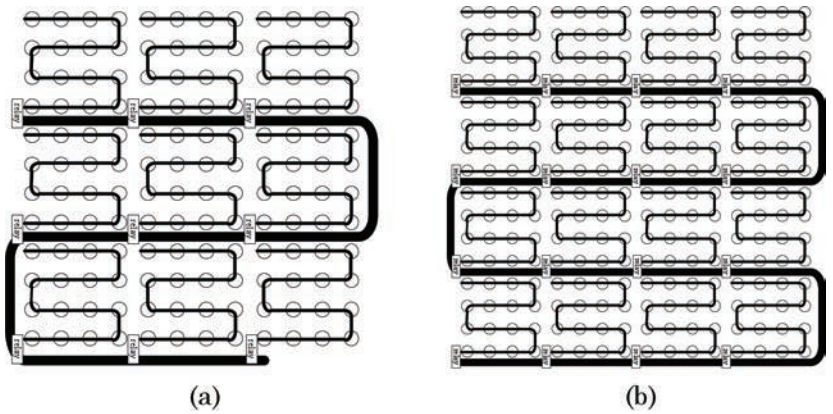

Fig. 16. Hierarchical architecture of (a) 12x12 RF nodes for a 24x24 router NoC; (b) 16x16 RF nodes for a 32x32 router NoC. Each 2x2 routers share one RF node.

As we scale up the number of components on chip to such a large level, full-system simulation becomes intractable due to extremely long runtime (on the order of weeks to months). Therefore, in this section we used a trace-driven cycle-accurate network simulator to evaluate performance and power consumption. We extended Garnet [Agarwal et al. 2009] to support the proposed hierarchical stream arbitration scheme. Moreover, as a means of exploring the interconnect demands of future applications, we made use of the probabilistic trace methodology developed in Chang et al. [2008a] to represent five communication patterns for multithreaded applications: uniform, uni-dataflow, bi-dataflow, 1hotspot and 2hotspot. To mimic the local-communication-dominated communication patterns expected in large-scale CMPs, our probabilistic traces have 60% of the communication as local traffic (inside the local TL). The remaining traffic follows the specific communication pattern as the trace name indicated: (1) uniform—the routers are equally likely to communicate with all other routers in different curls; (2) 1hotspot/2hotspot—there is one router in one/two TLs sending/receiving a disproportionate amount of traffic; (3) uni-dataflow/bi-dataflow—dataflow pattern simulates pipelined a communication flow such as medical imaging decomposition or a cryptographic algorithm, routers are biased to communicate with routers in groups that neighbor them on either one side (unidirectional dataflow) or two sides (bidirectional dataflow). The message sizes were either 8 bytes (a cache block request or control signal) or 64 bytes (a cache block response). Each probabilistic trace is executed on Garnet for 1 million network cycles.

6.3. Results

Figure 17 shows the average network latency reduction through hierarchical stream arbitration (denoted as *HStream*) compared to flat stream arbitration (denoted as *FStream*). By serving the local data channel requests in the local TL with much less latency, HStream arbitration can reduce the average network latency by 28% to 55% (40% on average) compared to FStream. The latency reduction in arbitration actually is similar for all of the five patterns (they all have the similar percentage of local traffic); however, there are still observable differences between the overall latency reduction of them. The reason for the considerable difference in performance observed has less to do with the arbitration latency, or even the latency of actual data transmission, but rather with the increased success of arbitration in the case of HStream as compared to FStream. The reason for this is intuitive: As the network increases, HStream maintains a constant number of competing nodes for the messages within the local TL.

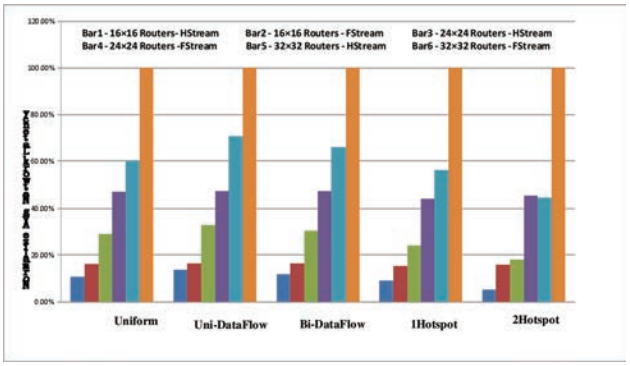

Fig. 17. Performance (average network flit latency) gain through hierarchical stream arbitration (normalized to flat stream of 32 × 32 topology for each application).

Communication occurring in disjoint regions of the NoC does not compete for a shared resource, where as in FStream it does. Even though in FStream this shared resource is of greater capacity, being the sum of the local TL and the global TL in terms of bandwidth, this doesn't make up for the large increase in competing nodes in a network that is not segregated into regions. Although in HStream there are two additional buffering steps in the relay nodes, experimental results show that most of these long- distance messages are granted data channels in the next arbitration immediately following the time they enter the relay node. We observe a reduction in the benefit of HStream as compared to FStream in the hot-spot experiments. The primary source of this reduced benefit is attributable to availability of buffer space in the node relay in transition point from the local TL containing a hot spot to the global TL. FStream does not exhibit this bottleneck, and thus performs similarly to its performance in other experiments.

Figure 18 shows the power comparison results of HStream and FStream, which are broken down into arbitration power and data transfer power. The data transfer power also includes the power between the network interface and the RF node. The reduced data transfer required to perform arbitration, as described in Section 6.1, allows HStream to reduce the energy required by arbitration by 40% to 70% compared to FStream. The larger the network size, the more significant is this arbitration reduction to the overall power reduction. However, in HStrean, all of the communications between different local TLs need to do modulation/demodulations for the buffer during transitions between the local TL and the global TL. As mentioned, this global communication only accounts for a small percentage of the overall communication in the large-scale CMPs; thus HStream only incurs a 3% to 16% additional data transfer power. The overall power of HStream and FStream is similar, within 5%.

7. RELATED WORK

To provide a guaranteed quality of service (QoS) in NoC in terms of throughput and latency, hybrid circuit switching and packet switching is introduced in recent work. The key is to dynamically construct virtual circuit. The first categories of work use Time-Division-Multiplexing (TDM). In a particular time interval, the available network bandwidth is exclusively dedicated to a virtual circuit. One of the representative works is the Philips Aethereal [Goossens et al. 2005] which uses two separate NoC: a Guaranteed Service (GS) circuit switching subnetwork and a Best-Effort (BE) packet-switching sub-network, where the BE network is used to configure circuit switching in the GS network. A similar hybrid NoC architecture is also used in the Intel Tile Processor [Wentzlaff et al. 2007]. TDM-based circuit-switching is particularly well adapted for long and

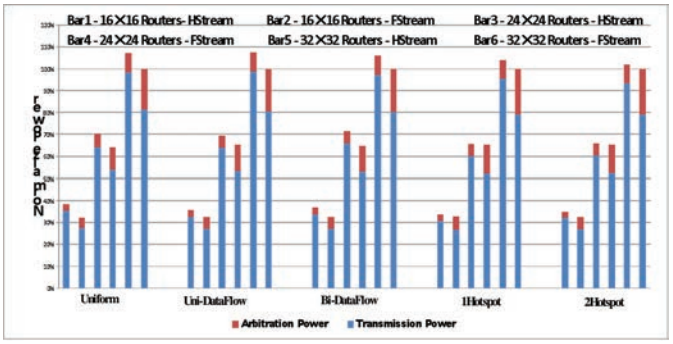

Fig. 18. Power savings through hierarchical stream arbitration (normalized to flat stream of 32×32 topology for each application).

frequent messages like multimedia streams. However, it suffers a long circuit set-up time and complex commutation scheduling. Recent work in Jerger et al. [2007] uses a hybrid router design that intermingles circuit-switched flits and packet-switched flits to reduce the circuit setup time. Spatial-Division-Multiplexing (SDM) is later introduced in Leroy et al. [2005] and enhanced by Modarressi et al. [2009] and Lusala and Legat [2010] to further reduce the circuit setup overhead by allocating a sub-set of the link wires to a given circuit for the whole connection lifetime. The proposed stream arbitration is similar to virtual circuit switching in the way of dynamically allocating communication channels between requested communication nodes. However, stream arbitration distinguishes the previous virtual circuit switching in that it circulates the information of available communication resources and the competing senders across all the arbitration participants, thus providing a much higher resource utilization and global fairness. Such information is impractical for circulating in the traditional RC-wire-based NoC, as it will incur significant latency and power overhead. Therefore, stream arbitration is not suitable for traditional RC-wire-based NoC. It is dedicated to the emerging high-bandwidth low-latency interconnects, such as RF-interconnect and optics.

To utilize the high bandwidth provided by emerging interconnects efficiently, researchers in Chang et al. [2008a] propose application-specific shortcuts that can be realized by dynamically tuning the on-chip RF transceiver frequencies. A shortcut is allocated based on the communication profiling of the application so that intensively communicated nodes will be allocated RF bandwidth. This approach is suitable for MPSoCs where the communication pattern is predictable or for applications with long and frequent messages like multimedia streams. However, it can not adapt well to the dynamic runtime variation in general-purpose CMPs. Token arbitration is proposed in Vantrease et al. [2008, 2009] to perform runtime arbitration of the communication channels by token-passing among the arbitration participants. Therefore, it works well for general-purpose CMPs with unpredictable dynamic variation behaviors. However, the communication bandwidth allocated to each receiver is fixed and may result in bandwidth waste when that receiver is not frequently used. The proposed stream arbitration distinguishes itself from Chang et al. [2008a] by allocating communication resource at runtime, and distinguishes itself from Vantrease et al. [2008, 2009] by allowing for any communication pairs to make use of communication channels in order to maximize the bandwidth utilization, which has been quantitatively proved in Section 5.

The curl-based arbitration channel used in stream arbitration is similar to the Intel ring-based communication architecture [Riedlinger et al. 2011], but with a much higher transmission latency and bandwidth because of the use of RF interconnect. Since it is difficult for the RF signal to be transmitted in a closed-loop ring due to

the impedance switching, we therefore use a curl style transmission line to allow the signal to take a round-trip of all the RF nodes. Moreover, the introducing of curl also allows us to pipeline the stream arbitration as the first trip is physically disjoint from the second trip.

8. CONCLUSION

In this work, we presented stream arbitration, a scheme for resource arbitration for emerging network technologies. We showed that our stream arbitration effectively utilizes resources of emerging network technologies, while being minimally invasive to circuit design. Additionally, we acknowledge that the NoC design will not scale in isolation, and demonstrated the effectiveness of our design in a simulated system consisting of components that are predicted to be central to future high-performance processor design. Our arbitration scheme efficiently deals with highly non-uniform traffic, while allowing high utilization. We presented a case study consisting of a modeled RF-I network, and compared this against an existing arbitration solution targeting emerging technology. Stream arbitration achieves an upwards of 40% reduction in average flit transmission latency, while making effective use of scarce network resources. Additionally, stream arbitration scales well with the number of communicating nodes, and accommodates both low- and high-traffic situations without degradation.

REFERENCES

Agarwal, N., Krishna, T., Peh L.-S., and Jha, N.K. 2009. GARNET: A detailed on-chip network model inside a full-system simulator, In *IEEE International Symposium on Performance Analysis of Systems and Software, (ISPASS 2009)*. 33–42.

Beckmann, B. M. and Wood, D. A. 2004. Managing wire delay in large chip-multiprocessor caches. In MICRO 37: In *Proceedings of the 37th Annual IEEE/ACM International Symposium on Micro Architecture*, IEEE Computer Society, pp. 319–330.

Bienia, C., Kumar, S., Singh, J. P., and Li, K. 2008. The PARSEC benchmark suite: Characterization and architectural implications. Tech. Rep. TR-811-08, Princeton University.

Chang, M. F., Cong, J., Kaplan, A., Liu, C., Naik, M., Prrvrumar, J., Retnman, G., Socher, E., and Tam, S. 2008a. Power reduction of CMP communication networks via RF-interconnects. In *Proceedings of the 41st Annual IEEE/ACM Internotional Symposium on Microarchitecture (MICRO 41)*. 376–387.

Chang, M. F., Cong, J., Kaplan, A., Naik, M., Reinman, G., Socher, E., and Tam, S.-W. 2008b. CMP network-on-chip overlaid with multi-band RF-interconnect. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. l9l–202.

Chang, M. F., Socher, E., Tam, R., Cong, J., and Reinman, G. 2008c. RF interconnects for communications on-chip. In *Proceedings of the 2008 International Symposium on Physical Design (ISPD '08)*. ACM, New York, 78–83.

Chang, M. F., Verbauwhede, I., Chien, C., Xu, Z., Kim, J., Ko, J., Gu, Q., and Lai, B. 2005. Advanced RF/baseband interconnect schemes for inter- and intra-ULSI communications. *IEEE Trans. Elect. Dev. 52*, 7, l27l–1285

Cho, S. and Jin, L. 2006. Managing distributed, shared L2 caches through OS-level page allocation. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 39)*, 455–468.

Cong, J., Ghodrat, M. A., Gill, M., Grigorian, B., and Reinman, G. 2012a. CHARM: A composable heterogeneous accelerator-rich microprocessor. In *proceedings of the International Symposium on Low Power Electronics and Design (ISLPED 2012).*

Cong, J., Ghodrat, M. A., Gill, M., Chunyue, L., and Reinman, G. 2012b. BiN: A buffer-in-NUCA scheme for accelerator-rich CMPs. In *proceedings of the International Symposium on Low Power Electronics and Design (ISLPED2012),*

Cong, J., Han, G., Jagannathan, A., Reinman, G., and Rutkowski, K. 2007. Accelerating sequential applications on CMPs using core spilling, *IEEE Trans. Paral. Distrib. Syst., 18*, 8, 1094–1107.

Cong, J., Liu, B., Neuendorffer, S., Noguera, J., Vissers, K. and Zhang, Z. 2011. High-level synthesis for FPGAs: From prototyping to deployment. *IEEE Trans. Comput.-Aided Desi. Integ. Circ. Syst.*, 30, 4, 473–491.

Cong, J., Liu, C., and Reinman, G. 2010. ACES: Application-specific cycle elimination and splitting for deadlock-free routing on irregular network-on-chip. In *Proceedings of the 47th Design Automation Conference (DAC)*. 443–448.

Constantinou, T., Sazeides, Y., Michaud, P., Fetis, D., and Seznec, A. 2005. Performance implications of single thread migration on a chip multi-core. *SIGARCH Comput. Archit. News 33*, 4, 80–91.

Duato, J. and Pinkston, T. M. 2001. A general theory for deadlock-free adaptive routing using a mixed set of resources. *IEEE Trans. Paral. Distrib. Syst.*, *12, 12*, 1219–1235.

Goossens, K., Dielissen, J., and Radulescu, A. 2005. AETHEREAL network on-chip concepts. *IEEE Desi. Test comput., 22*, 5, 414–421.

Hardavellas, N., Ferdman, M., Falsafi, B., and Ailamaki, A. 2009. Reactive NUCA: Near-optimal block placement and replication in distributed caches. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA '09).* ACM, New York, 184–195.

Jerger, N. E., Peh L-S., and Lipasti, M. 2007. Circuit-Switched Coherence. In *computer architecture letters, 6*, 1, 5–8.

Kahng, A., Li, B., Peh, L.-S., and Samadi, K. 2009. ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2009)* 423–428.

Kim, Y., Byun, G.-S., Tang, A., Jou, C.-P., Hsieh, H.-H., Reinman, G., Cong, J., and Chang, M. F. 2012. An 8Gb/s/pin 4pJ/b/pin single-t-line dual (Base+RF) band simultaneous bidirectional mobile memory I/O interface, In *proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)* 50–51.

Kumar, R., Zyuban, V., and Tullsen, D. M. 2005. Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA '05).* IEEE Computer Society, 408–419.

Lee, H., Cho, S., and Bruce R.C. 2010. StimulusCache: Boosting performance of chip multiprocessors with excess cache. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA),* 211–222.

Lee, H., Cho, S., and Bruce R.C. 2011. CloudCache: Expanding and shrinking private caches. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA).* 219–230.

Leroy, A., Marcnet, P., Shickova, A., Catthoor, F., Robert, F., and Vertest, D. 2005. Spatial division multiplexing: A novel approach for guaranteed throughput on NoCs. In *Proceedings of the 3rd IEEE/ACIWIFIP International Conference on Hqrdware/Software Co-Design and System Svnthesis*. 8l–86.

Lusalsa, A. K. and Legat, J.-D. 2010. A hybrid NoC combining SDM-based circuit switching with packet switching for real-time applications. *NORCHIP*, 15–16, Nov, 1–4.

Magnusson, P., Christensson, M., Eskilson, J., Forgren, D., Hallberg, G., Hogberg, J., Larsson, F., Moestedt, A., and Werner, B. 2002. SIMICS: A full system simulation platform. *IEEE Computer, 35*, 2, 50–58.

Martin, M., Sorin, D., Beckmann, B., Marty, M., Xu, M., Alameldeen, A., Moore, K., Hill, M., and Wood, D. 2005. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset, *In Comput. Archi. News*, *33*, 4, 92–99.

Modarressi, M., Sarbazi-Azad, H., and Arjomand M. 2009. A hybrid packet-circuit switched on-chip network based on SDM. In *Proceedings of the Conference on Design, Automation and test in Europe (DATE'09)*. 566–569.

Qureshi, M. and Patt, Y. 2006. Utility-based cache partitioning: a low-overhead, high-performance, runtime mechanism to partition shared caches. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 39),* 423–432.

Riedlinger, R. J., Bhatia, R., Biro, L., Bowhill, B., Fetzer, E., Gronowski, P., and Grutkowski, T. 2011. A 32nm 3.1 billion transistor 12-wide-issue Itanium® processor for mission-critical servers. In *proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 84–86.

Tam, S.-W., Socher, E., Wong, A., and Chang, M. F. 2009. A simultaneous tri-band on-chip RF-Interconnect for future Network-on-Chip, In *proceedings of the IEEE VLSI Symposium.* 90–91.

Vantrease, D., Schreiber, R., Monchiero, M., Mclaren, M., Jouppi, N. P., Fiorentino, M., Davis, A., Binkert, N., Beausoleil, R. G., and Ahn, J. H. 2008. Corona: System implications of emerging nanophotonic technology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA '08).* IEEE Computer Society, Washington, DC, 153–164.

Vantrease, D., Binkert, N., Schreiber, R., and Lipasti, M. H. 2009. Light speed arbitration and flow control for nanophotonic interconnects. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 42).* ACM, New York, 304–315.

Wentzlaff, D., Griffin, P., Hoffmann, H., Bao, L., Edwards, B., Ramey, C., Mattina, M., Miao, C-C., Brown, J. F., and Agarwal, A. 2007. On-Chip Interconnection Architecture of the Tile Processor, *Micro, IEEE 27*, 5, 15–31.

WU, H., NAN, L., TAM, S.-2., HSIEH, H.-H., JOU, C., REINMAN, G., CONG, J., AND CHANG, M.-C. 2012. A 60GHz on-chip RF-interconnect with $\lambda/4$ coupler for 5Gbps bi-directional communication and multi-drop arbitration In *Proceedings of the IEEE 34th Custom Integrated Circuits Conference*.

Received June 2012; revised September 2012 and November 2012; accepted November 2012