Towards Scalable, Energy-Efficient, Bus-Based On-Chip Networks [*]

Aniruddha N. Udipi
University of Utah
udipi@cs.utah.edu

Naveen Muralimanohar
HP Labs
naveen.muralimanohar@hp.com

Rajeev Balasubramonian
University of Utah
rajeev@cs.utah.edu

Abstract

It is expected that future on-chip networks for many-core processors will impose huge overheads in terms of energy, delay, complexity, verification effort, and area. There is a common belief that the bandwidth necessary for future applications can only be provided by employing packet-switched networks with complex routers and a scalable directory-based coherence protocol. We posit that such a scheme might likely be overkill in a well designed system in addition to being expensive in terms of power because of a large number of power-hungry routers. We show that bus-based networks with snooping protocols can significantly lower energy consumption and simplify network/protocol design and verification, with no loss in performance. We achieve these characteristics by dividing the chip into multiple segments, each having its own broadcast bus, with these buses further connected by a central bus. This helps eliminate expensive routers, but suffers from the energy overhead of long wires. We propose the use of multiple Bloom filters to effectively track data presence in the cache and restrict bus broadcasts to a subset of segments, significantly reducing energy consumption. We further show that the use of OS page coloring helps maximize locality and improves the effectiveness of the Bloom filters. We also employ low-swing wiring to further reduce the energy overheads of the links. Performance can also be improved at relatively low costs by utilizing more of the abundant metal budgets on-chip and employing multiple address-interleaved buses rather than multiple routers. Thus, with the combination of all the above innovations, we extend the scalability of buses and believe that buses can be a viable and attractive option for future on-chip networks. We show energy reductions of up to 31X on average compared to many state-of-the-art packet switched networks.

1. Introduction

The future of multi-core computing is likely to diverge into two separate tracks targeting different application domains. On one hand, we will likely see mid-range processors with a moderate number of cores (say 32 or 64), targeted at home/office/light computation. On the other hand, we will see high-end processors with much larger core-counts (in the hundreds, if not more), targeted at large server/scientific/commercial applications. These are most likely to be used in a virtualized environment, with small subsets of cores being assigned to different virtual machines, running independent applications for different users. As a result, on-chip communication will likely be localized within a group of cores (say, 64 cores) with a few workloads requiring non-trivial communication between groups. This argues for hierarchical network topologies, such as the one shown in Figure 1. The processor has 1024 cores organized into 16 groups. A grid network connects the 16 groups and a grid topology is also employed within each group of 64 cores (such a hierarchical topology bears similarities to a topology with express channels). The work in this paper focuses on the design of the intra-group on-chip network for 16-64 cores as this will likely form an important unit in both mid-range and many-core processors. The design of such a network has been actively studied in recent years, but there is yet little consensus on what constitutes an optimal network [32].

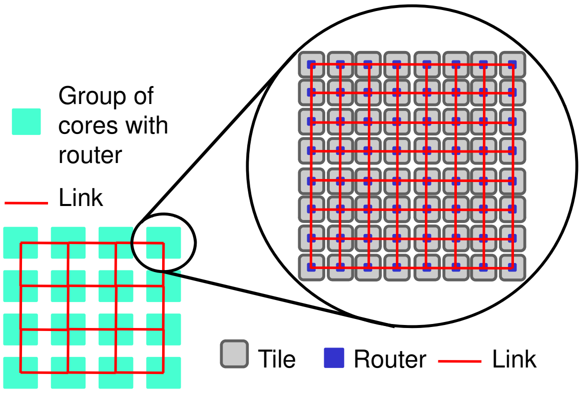


Figure 1. Hierarchical interconnect structure with grid topology (and directory-based cache coherence)

On-chip networks can represent a significant bottleneck in terms of energy, delay, complexity, verification effort, and area [15]. Many recent papers have examined on-chip network implementations that incorporate many features in an attempt to eke out high performance. Much of this research area has been heavily influenced by Dally and Towles' seminal paper [16] that argued for the use of on-chip packet-switched networks as a means to provide high bandwidth. As a result, in most NoC papers that attempt to connect more than eight devices, it is common to assume a grid/torus topology with wormhole routing and virtual channel flow control [17]. Indeed, this approach has been

[*] This work was supported in parts by NSF grants CCF-0430063, CCF-0811249, CCF-0916436, NSF CAREER award CCF-0545959, Intel, SRC grant 1847.001, and the University of Utah.

commercially successful in the 20th century when building large scale multi-processors with off-chip inter-processor wiring. Many researchers have assumed that many interconnection network design principles from the 20th century will continue to be directly applied to on-chip networks. For example, one automatically assumes that a system with 16 or more on-chip cores will employ a packet-switched grid network and a directory-based cache coherence protocol. There are, however, two important disadvantages to such a design. First, a directory based protocol inherently suffers from indirection, requiring multiple messages for every coherence transaction. This in general means more time and energy consumption for the operation to complete. Second, every one of these messages has to traverse multiple routers in the network, which entails significant drawbacks. Traversal through most routing elements requires multiple clock cycles (example commercial router pipeline depths are four cycles [30] and eight cycles [36]), significantly impacting execution time. More importantly, routers consume huge amounts of power, even orders of magnitude more than a single link traversal (actual numbers and methodology follow in Section 3). This is likely to be a huge problem as energy consumption becomes a more important criterion in system design. In fact, projections from Intel [30] indicate that such on-chip networks can contribute 20-36% of total chip power (depending on whether the network connects dense compute units or cache banks).

This paper is an attempt to re-visit some of the basic observations that drive interconnection network design. The work was inspired by a controversial view expressed by Shekhar Borkar at ISLPED '07 and the OCIN Workshop '06 [7, 8]. In these talks, Borkar argues that metal/wiring are cheap and plentiful. Wire delays are somewhat tolerable when repeaters are judiciously employed. The use of repeaters also enables easy pipelining. These are properties that did not exist for off-chip wiring. We are also no longer pin-limited since we are connecting multiple cores on a die and not going off-chip. Borkar further argues that the elimination of routers and the complex functionalities in them will contribute greatly to alleviating the power and verification problem. We thus attempt to extend the life of the broadcast bus and explore ways to allow it to scale to a moderately large number of cores (16-64).

By eliminating routers, the overheads of communication are purely because of wiring, bringing us closer to the *ideal interconnect fabric* outlined by Kumar et al. [28], although with a very different approach. The primary problem with completely eliminating routers is that every coherence transaction now has to be broadcast across the entire chip. This poses a significant energy overhead as the number of cores on die scales. It is well known that a majority of snoop requests are ultimately not useful because most cores don't have the relevant cache lines anyway [35, 44], wasting energy on every transaction. It is thus possible to get closer to the ideal fabric in terms of power and performance by somehow making sure the broadcast is only seen by relevant nodes. We propose to do this by segmenting the chip into multiple segments, each with its own broadcast sub-bus. Entry and exit of messages in these segments is governed by bloom filters that track the presence of data in various cores. If a majority of traffic is local, or is forced to be local by smart page coloring, we greatly reduce the number of links that need to get activated on every transaction. Now that the network is link-dominated, it is also possible to significantly impact its energy characteristics by employing low-swing wiring for the links. Such an optimization would be far less impactful in a packet-switched network that is router-dominated.

It is also well-known that snooping-based coherence protocols are easier to design and verify. Buses and arbiters are also easier to design and verify than the many components within a router. The proposed approach also frees the interconnect fabric from having to worry about deadlock avoidance and congestion control. Hence, our findings have favorable implications on the design process: in our efforts to optimize energy, we have also improved ease of design. This further motivates industry to stay on the bus-based path for longer than initially anticipated. Clearly, as we continue to scale up the number of cores, there is a point where bus-based snooping will fail, perhaps arguing for hierarchical coherence protocols beyond that point.

The contributions of our paper are six-fold: First, we propose the use of a bus and snooping protocols as an attractive alternative to packet-switched networks in energy constrained architectures. Second, we propose a novel filtered segmented bus organization that removes the inefficiency associated with chip-wide broadcasts in snooping-based protocols. Third, we employ OS-based page coloring to increase locality, improve the efficiency of our filters and further reduce network energy, showing that this is an especially compelling optimization in bus-based networks. Fourth, we propose the use of multiple address-interleaved buses as a way to reduce contention and improve performance while utilizing the abundant metal budgets available on chip. Fifth, we use low-swing wiring to reduce the energy overheads of long bus links. With the combination of all the above innovations, we extend the scalability of the bus as a viable and attractive option for future on-chip networks. Finally, we determine the extent of router optimization required to enable packet-switched networks to match the energy characteristics of bus-based networks.

The rest of the paper is organized as follows: Section 2 details the various network organizations we consider. We present methodology and results in Section 3, discuss related work in Section 4 and conclude in Section 5.

2. Interconnect Structures

We consider a chip multiprocessor where each core has a private L1 cache and logically shares a large L2 cache. The L2 cache is physically distributed on chip, with one slice associated with each core (essentially forming a tiled chip). Sets are distributed among these slices in an S-NUCA organization [24]. OS-based page coloring can help map a given page to a specific slice. Coherence is maintained among the L1 caches and the L1-L2 hierarchy is assumed to be inclusive. We consider snooping or directory-based protocols, depending on the interconnect structure. We explore a number of different organizations for the interconnect fabric.

2.1 Baseline Packet Switched Networks

2.1.1 Grid Network

The first baseline organization (that represents the de facto standard for most scalable evaluations) is the conventional grid network with directory-based coherence. An example network for a tiled CMP is shown in Figure 1, with 64 cores forming an 8x8 grid. Each core is associated with a 5x5 router that can send or receive packets from all four directions and also from its own core. Since each link of the grid network allows a different message to be in transit, it is generally assumed that such a structure provides high scalability, high bandwidth, high degree of concurrency between coherence transactions, and hence high performance. The primary disadvantages are home directory indirection (leading to multiple messages per transaction), significant router overheads (energy, delay and verification), and complexity of directory-based protocols (leading to increased design costs).

2.1.2 Ring Network

Commercial designs, especially those from Intel, are moving towards a ring interconnect for future multi-cores [43]. They are relatively simple to design and route on chip. An example architecture is shown in Figure 2(a). A big win with the ring network is the use of relatively low-radix 3x3 routers, which consume lower energy than their counterparts in the grid or high-radix networks, since energy increases super-linearly with radix, per Orion 2.0 [23]. This, however, comes at the cost of increased network diameter.

We use two simple optimizations to improve the performance of the baseline ring topology. We assume a directory based protocol and stop the ring transaction when all relevant nodes (as indicated by the directory) have seen the message, saving energy by not traversing the entire ring on every transaction. Second, we use both a clockwise and a counter-clockwise ring, picking the one with the least distance that needs to be covered to reach all destinations.

2.1.3 Flattened Butterfly

The flattened butterfly architecture was proposed by Kim et al. [25] to exploit high-radix routers on chip. Every router is directly connected not only to its immediate neighbors but also to every router in its own row and column as shown in Figure 2(b). For clarity, we only show one such heavily connected node. For this study we assume a concentration factor of just one, in an attempt to reduce router radix. This architecture reduces the diameter of the network and the number of hops required for every message. It also addresses to some extent the issue of multiple cycle router traversals by amortizing those latencies over longer distances covered on chip. The downside, however, is that high-radix routers are power hungry. While optimizations have been proposed to address these overheads [25, 26, 46], they may still not be good fits in power-constrained architectures. They also complicate the wire routing process on-chip.

2.2 Proposed Bus-based Networks

Conventionally, bus design assumes one large monolithic bus connecting all the cores on the die. Every transaction is broadcast on this bus sequentially, with each core maintaining coherence by continuously monitoring the bus.

2.2.1 Shorted Bus

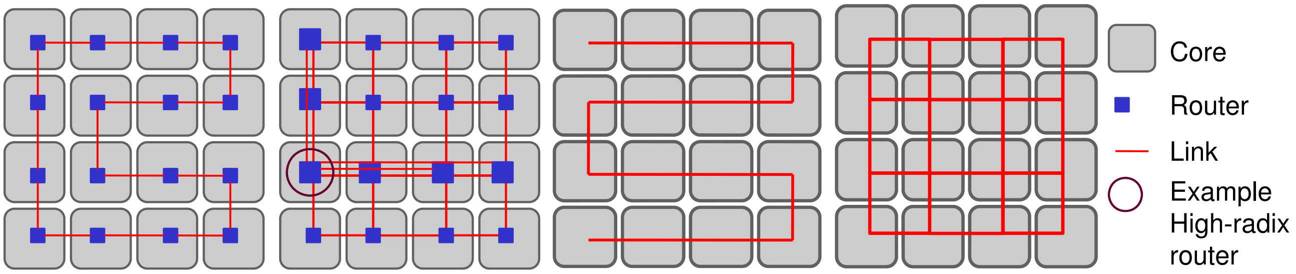
As a first attempt at improving the characteristics of the conventional bus (Figure 2(c)), we tailor the physical layout of the bus for tiled organizations, electrically shorted all around as shown in Figure 2(d). The bus fans out at various points to reduce the overall distance that must be traversed, with the longest latency ideally being just the Manhattan delay between two opposite corners of the chip. There is a also a repeater at every tile to reduce delays and allow operation at a high frequency. Since repeaters impose directionality on a link, each link is composed of two sets of wires, each heading in opposite directions. Such a bus has two major advantages: first, unlike the conventional bus, it is latency-optimized and obviously improves performance since each transaction holds up the bus for fewer cycles. Second, as in any bus, it inherently avoids ordering issues between coherence transactions.

To arbitrate for the bus, a tile must send a request signal to a central arbiter structure that then sends back a grant signal to one of the requesters. Even as the number of nodes scales up, this will likely not represent a bottleneck in terms of wiring as only two wires are required per node. If the worst-case delay for broadcast on the bus is D cycles, the arbiter grants access to one of the nodes every D cycles and appropriately schedules the grant signal (based on the delay for the grant signal to reach that tile). Note that this arbiter latency is on the critical path in lightly-loaded conditions but is a non-issue if the bus is well-utilized. To keep the arbiter design simple, we assume that each node only has one outstanding bus request, the request signal is activated until a grant is received, and there is no buffering of requests at the arbiter. When a tile receives a grant signal, the coherence request is placed on the *address bus* and it propagates in both directions over the next D cycles. For a high-performance split-transaction bus design, the next request can be placed on the address bus after D cycles, while the coherence responses are handled on a separate *control and data bus*. To recap, the primary advantages of the simple bus are: no indirection when handling coherence requests, elimination of routers and easier protocol verification, at the cost of long wire traversals.

2.2.2 Segmented bus

While the shorted bus described above is simple to design and has its advantages, transactions are necessarily serialized, causing heavy contention and poor performance. In an effort to improve these characteristics of the bus, we propose a hierarchical *segmented bus* structure. We propose dividing the chip into several segments of cores, with a *shorted bus* connecting every core in a segment. Each of these *"sub-buses"* is then connected to a central bus as shown in Figure 3. Simple tristate gates are present at the intersection of every sub-bus with the central bus to enable messages to move from one bus to the other.

Since every transaction still has to be broadcast on all sub-buses (and the central bus), we retain the same global arbitration as in the single shorted bus. In this organization, however, the arbiter looks for three consecutive cycles *i*, *i+1* and *i+2* where the originating sub-bus, central bus and remote sub-buses respectively are free for broadcast. At



(a) Ring network (b) Flattened butterfly network (c) Conventional bus (d) Shorted bus

Figure 2. **Example baseline networks**

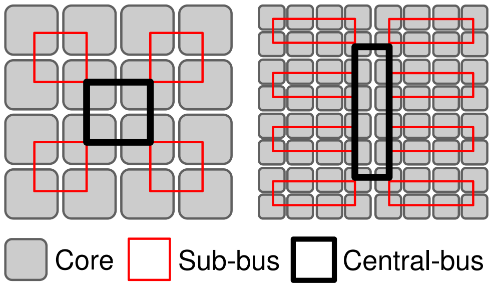


Figure 3. **Segmented Bus structure shown for 16 and 64 core processors**

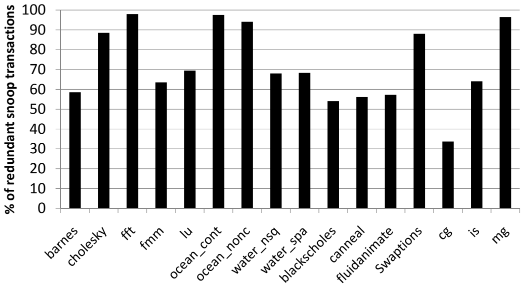


Figure 4. **Redundant snoop transactions**

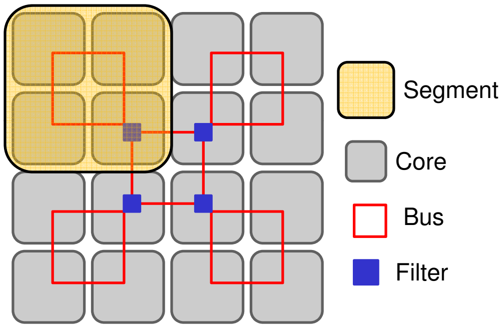


Figure 5. **Filtered segmented bus**

this time, the entire broadcast is completed, avoiding the need for any buffering of messages at intermediate locations along the network.

This design essentially allows every bus transaction to now be a 3-stage pipeline, consisting of the originating sub-bus broadcast, central-bus broadcast and remote sub-bus broadcasts. In addition to increasing throughput, this also reduces contention for the buses, giving a small performance boost. Note that two successive broadcasts are pipelined and happen in back-to-back cycles only if they originate from the same sub-bus. If not, the start of two transactions is separated by 3 cycles. Also, requests from the central bus are always given priority over local requests in any sub-bus. This ensures that once the central bus is acquired, the remote buses are available in the consecutive cycle. The central bus thus acts as the serialization point and enforces coherent ordering of transactions.

We continue to assume a split-transaction bus with data messages being handled separately. Data message transactions can be further optimized since source and destination are known a priori. If it is known that the requester and responder are in the same segment, only that local data sub-bus is activated. If not, the requester sub-bus, the responder sub-bus and the central bus are arbitrated for, and activated in consecutive cycles to complete the data transfer.

2.2.3 Filtered segmented bus

While both bus structures discussed in the previous subsections eliminate routers and the associated power overheads, the energy dissipated simply by charging and discharging long links is a deal breaker. Unlike packet switched networks, *every* link is activated on every transaction, which is certainly not effective as the number of cores scales.

Figure 4 shows the fraction of snoop transactions that were redundant i.e. no core needed to take action on them in a MESI protocol (details on methodology appear in the next section). It is clear from this figure that for a large number of coherence transactions, very few nodes need to see the broadcast at all, making a naive snooping implementation inefficient. We therefore propose a novel *filtered bus* architecture (Figure 5), where we maintain some knowledge of cache contents in the local and remote segments, and use this data to stifle broadcasts as early as possible. We now describe the working and arbitration scheme for such a network.

Since not every transaction is now expected to be broadcast on every sub-bus, it would be inefficient to use global arbitration and reserve all buses in advance. We therefore do it in multiple steps, as and when required. On initiating a transaction, the requesting node first arbitrates for its local sub-bus and completes this local broadcast alone. This broadcast also serves the purpose of getting the address under consideration to a bloom filter in the segment which decides if the transaction needs to leave the segment and be seen by a core outside the segment (more details follow). If not, the bloom filter immediately "validates" the broadcast (via wired-OR signals [14]) and the transaction is deemed complete. If it does need to go out, the message then arbitrates to get onto the central bus and thereafter to the remote buses. The transaction is validated only after the central bus is acquired. At the remote sub-buses, bloom filters are consulted once again to see if a broadcast needs to occur in that segment (more details follow). Arbitration is begun only if required. Messages are thus only broadcast on a subset of segments on every transaction, and this stifling saves significant amounts of link energy.

Another advantage with such a scheme is that multiple nodes can now initiate and complete transactions simultaneously if the requests are not going past the local sub-bus. If the requests are to the same cache line and require serialization, this happens at the central bus. The order in which the transactions appear on the central bus is the order in which they will be seen by all cores.

Since the various buses operate independently with decentralized arbitration, careful design is required to ensure correct operation. For instance, assume that a core A in segment 1 wants to start a transaction in cycle *i*. Also assume that a core B in segment 2 started a transaction a few cycles earlier that crossed the central bus and wants to get on the sub-bus in segment 1, also in the same cycle *i*. Since B received ownership of the central bus first, its transaction must go through first and the local sub-bus arbiter must ensure that requests originating in the central bus are given priority over requests originating within the segment. Also, some small buffering is required to hold the message from A while it is waiting for access to the central bus. In the absence of such buffering, messages would have to be broadcast once to consult the filters and if the central bus is unavailable, the broadcast will have to be canceled and re-transmitted, wasting energy. This design also resolves any potential deadlock situation where A has completed its local broadcast and is waiting for the central bus (occupied by B, say) and B has completed its central broadcast and is waiting for A's sub-bus.

Note that one potential problem with such a scheme is starvation of local broadcasts if external messages keep coming in. This can be handled by having two separate sub-buses, one reserved exclusively for incoming global transactions that is activated a little ahead in time of the regular local sub-bus in every cycle. However, our experiments show that this is not a major bottleneck to good performance.

An alternative to the buffered and decentralized arbitration described above would be to have dedicated address lines from every core to the bloom filter in the segment. Depending on the bloom filter results, either local or global arbitration can be completed to allow the message to simply be broadcast without any conflicts or buffering. However, this would significantly increase wiring overheads in addition to increasing the number of bit transfers required on every transaction. Further, it would require communication between the global and local arbiters to allow correct scheduling of transactions. Our decentralized scheme avoids these overheads and is therefore more efficient.

There is also a difference in the operational latencies of the segmented bus with and without filtering. In the hierarchical segmented bus, there is one global arbitration, with a 14 cycle round-trip delay for the request/grant signal propagation (corresponding to a trip halfway across the chip and back, more details are in Section 3). This is followed by 12 cycles of broadcast (4 cycles each on the local sub-bus, central bus and remote sub-buses), resulting in a 26 cycle latency overall. As with the shorted bus, the request/grant signals are on the critical path when there is little or no load, else they can be overlapped with previous bus broadcasts. In the filtered bus, the initial broadcast requires a local arbitration of 4 cycles for the local sub-bus (again, the round-trip delay to go halfway across the segment, which may or may not overlap with a previous broadcast). This is followed by a 4 cycle broadcast and a 1 cycle bloom filter lookup. If the bloom filter indicates that the transaction is over at this point, the total latency for the transaction is just 9 cycles. If not, the same latencies repeat at the central bus, adding another 9 cycles (this time checking the bloom filters to enter the remote segments). If the bloom filters indicate that the message need not be broadcast in the remote segments (it was a false positive in the outgoing bloom filter), the total latency is now 18 cycles. In the worst case, if broadcasts are required in the remote segment(s) too, there is an additional 8 cycle latency, bringing the total to 26 cycles, which is the same as the segmented bus. However, since many broadcasts are going to be stifled after just one local broadcast or after the central bus broadcast, the average latency is lowered significantly.

The filtered segmented bus design retains all the advantages of the previously described segmented bus, and adds further innovations to improve the energy and performance characteristics of the bus.

2.2.4 Stifling snoop broadcasts

Rather than keep track of every line in the cache individually, we track presence of data in the cache using signatures and bloom filters. We define the *"home-node"* of a line as being the slice of L2 in which it resides, either with a default S-NUCA placement policy or an OS-assisted page-colored placement. We maintain two filters per segment, which we call the *Out-filter* and the *In-filter*.

The *Out-filter* in each segment keeps track of "local" cache lines (lines calling that segment home) that have been sent *out* of that segment i.e. requested by a core outside that segment. This is useful when a core in a segment broadcasts, say, an invalidate. If that particular line was never sent out of that segment, there is no need to broadcast it beyond the local segment. Note that if the "home node" of the line is in a different segment, the broadcast still has to go on the central bus, though it may be stifled later. The *In-filter* keeps track of cache lines currently present in a segment. When a broadcast, say an invalidate, appears on the central bus connecting the filters, every segment filter looks up the line and broadcasts within its segment only if required. It must be stressed that in either case, we can never have false negatives since that would violate correctness of execution. We implement counting bloom filters to enable us to remove elements from the filter when a line is evicted or invalidated. Without this, the filters would quickly saturate and be ineffective. We can therefore no longer have silent evictions and we model the additional energy overhead because of this. The size and accuracy of the bloom filters determine the percentage of useless snoops that are effectively stifled, saving energy.

The design parameters of our bloom filters were determined empirically. Every bloom filter consists of 2 arrays of 8192 entries each. A simple mod function hashes into the array and increments/decrements a 10-bit counter. Each filter thus requires approximately 16 KB of storage. The energy and delay of accessing this structure are accurately modeled in our experiments.

2.2.5 Page coloring

A simple and yet effective way of increasing locality and ensuring that a majority of transactions don't have to go out of their local segment is to do OS-assisted page coloring for the L2 cache [13]. We perform *First-touch* page coloring, where the first node to touch a page becomes its assigned slice of L2. Several proposals have addressed more optimal implementations of page coloring [3, 10] but a detailed study of such techniques is beyond the scope of this paper. Note that while any network will be negatively impacted by sub-optimal placement of shared pages, the "segment" granularity of our bus-based network somewhat mitigates this problem by allowing for more flexibility in the placement of shared pages. If the number of accesses satisfied in a core's local L2 slice or local segment increases, and if the filter is effective in capturing this behavior, fewer broadcasts go out, saving significant amounts of energy.

2.2.6 Low-swing buses

Low-swing signaling is an effective way to reduce energy consumption in links by reducing the voltage range through which the wires are charged/discharged [21]. While utilizing low-swing wires will reduce the link energy in any on-chip network, they are an even more compelling choice for bus-based networks. Application of Amdahl's Law tells us that the returns of focusing on the link energy in a router-dominated packet-switched network will be much lower than in a link-dominated bus-based network.

2.2.7 Multiple address interleaved buses

It is expected that beyond a certain number of nodes, the fully shared nature of buses will likely be a significant performance bottleneck. To reduce contention pressure on the bus and increase concurrency, we exploit the abundant metal bandwidth present on-die. We stress that unlike traditional SMPs, we are no longer pin-limited. We therefore consider employing two parallel buses, interleaved by address. The two buses handle mutually exclusive addresses, effectively avoiding any ordering issues. This approach increases concurrency and bandwidth in the system while incurring a cost in terms of metal/wiring area/power, but without introducing the overheads of complex protocols and routing elements. Note that additional buses incur no dynamic energy overhead but will incur some leakage overhead. However, this is likely to be less than the leakage introduced by the buffers in an over-provisioned packet-switched network. A natural extension to such a scheme would be to increase the number of buses employed. We leave this exploration to future work.

3. Results

3.1 Methodology

We model a baseline 16-core tiled CMP with each tile consisting of a core, its private L1 cache and one slice of a shared L2 cache (see Table 1 for details). We retain the same latency and energy parameters for our 32- and 64-core experiments and assume an appropriate increase in die size. Coherence among the L1 caches is maintained using a MESI snoop protocol for the bus-based networks and a distributed MESI directory protocol for the packet switched networks. Our bus-based network models include detailed contention modeling to get bus access since this is typically considered an important bottleneck in bus-based systems. We also include the round-trip time required for the request/grant signals to travel to and from the arbiter as described in the previous section. To speed up our simulations, we model "ideal" packet-switched networks with accurate link and router latencies but ignore contention within the network. This gives an advantage to the baseline packet-switched networks that we are comparing our bus architectures against. This approximation does not affect energy numbers since packets have to traverse a fixed number of routers and links at some point regardless of contention, assuming deterministic routing.

As a simulation infrastructure, we use Virtutech's SIMICS [1] full system simulator with the 'g-cache' module (Simics' in-built cycle accurate cache model with MESI coherence) significantly modified to include various network architectures and latencies. We model low-swing interconnects [21, 22] for the links (in both bus-based and packet-switched networks) and a 3-cycle router at every node. Note that while commercial router pipelines are of the order of 4-8 cycles [30, 36], speculative routers have been proposed [17, 34, 37, 40] and are expected to be part of future on-chip networks. Wire and router energy/delay estimates are obtained using a combination of CACTI 6.0 [38] and Orion 2.0 [23]. Note that CACTI's router model borrows heavily from ORION 2.0. Wire energy numbers were also verified against estimates from the Orion group [11] and found to be in the same ballpark. These are shown in Table 1.

We evaluate our proposals on subsets of the PARSEC [6], NAS [4] and SPLASH-2 [49] benchmark suites and report execution time and energy consumption for the entire parallel sections of the workloads. We use the 'simlarge' configuration for the PARSEC suite, 'W' for the NAS suite and problem sizes larger than base size for the SPLASH-2 suite (see Table 2).

3.2 Results

We compare various characteristics of the proposed bus-based schemes and against three baseline packet-switched networks: the grid, ring and flattened butterfly. We first discuss the energy savings obtained in the bus-based networks, followed by a study of the performance impacts of such schemes and their ability to scale to larger core counts. We also derive the extent to which router energy will have to be optimized to make them feasible in future on-chip networks.

Die parameters	10 mm x 10 mm, 32nm,3GHz
L1 cache	Fully Private, 3 cycle 4-way, 32 KB Data 2-way, 16 KB Instr
L2 cache	Fully shared, unified S-NUCA 8-way, 32 MB total, 2 MB slice/tile 16 cycles/slice + Network delay
Main memory latency	200 cycles
Router	4 VCs, 8 buffers/VC, 3 cycles

(a) General parameters

Component	Energy (J)
2.5 mm low-swing wire	3.02e-14
2.5 mm full-swing wire	2.45e-13
3x3 ring router	7.32e-11
5x5 grid router	1.39e-10
7x7 flattened butterfly router	2.24e-10
Tristate gates (64)	2.46e-12
Bus arbiter	9.85e-13
Bloom filter	4.13e-13
Single-entry buffer	1.70e-13

(b) Energy numbers

Table 1. Methodology and energy parameters for our baseline 16-core model

Barnes	Cholesky	FFT	FMM
128K bodies	tk29.O	256K pts	32K particles
Ocean-Cont	Ocean-NonC	Water	Lu-Cont
1024 pts	256 pts	512 mol	1024x1024

Table 2. Problem size used for SPLASH-2 benchmarks

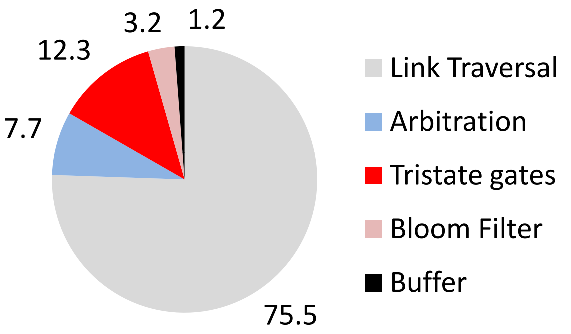


Figure 6. Contribution of various components to energy consumption in a broadcast

3.2.1 Energy Characteristics

Moving from a single shorted bus (Fig 2(d)) to a segmented bus (Fig 3) is not an energy optimization since we have to send every message to every node anyway. Energy can only be saved by introducing filters to stifle broadcasts as quickly as possible, activating the fewest possible number of segment buses.

Filtered segmented bus network: There are four major contributors to energy consumption on every broadcast: arbitration, link traversal, bloom filter access and the tristate gates to move between buses. The relative contributions of these to the total energy consumption remain fairly constant across benchmarks and the average values are shown in Figure 6. Figure 7 shows the relative energy consumed by our proposed filtered segmented bus when compared to the hierarchical bus and packet switched networks like the grid, ring and flattened butterfly. We see that even the most energy efficient packet switched network consumes an average of 20X as much energy as the segmented filtered bus, with a best case reduction of over 40X in the address network. In the data network, we see an average of 2X energy reduction compared to the most efficient packet-switched network with a best case reduction of 4.5X. Note that the data network energies of the segmented filtered bus and the hierarchical bus are practically the same and the minor fluctuations seen in this regard are simply due to the inherent non-determinism in multiple runs of the same parallel programs.

The large energy difference between bus-based and packet-switched networks can be explained as follows: Our experiments show that in a typical grid based packet switched network, an average of 2.5 hops are required per message, with a full-fledged router traversal at every hop. A single router traversal can consume up to 7X as much energy as a simple link traversal, leading to significantly increased energy consumption. This gap is further widened by the use of low-swing interconnects to up to 60X (detailed numbers are provided in Table 1(b)). The elimination of routers is thus an automatic win in an energy-constrained world.

Bloom filter efficiency: Figure 8 shows the accuracy of our stifling *In-* and *Out- bloom filters*. We see that the out-filters are accurate about 85% of the time on average. With this level of accuracy, we find that on average, about 30% of all broadcasts are stifled by the Out-filter and stay entirely local. The remaining 70% go out on the central bus and lookup the In-filters in the remote segments. Out of these, about 70% are completely stifled and not broadcast on any remote segment. About 20% get broadcast on one segment, about 6% on two and only 4% of all messages get broadcast on all three remote segments. The in-filters have a false positive rate of about 10% on average, effectively curbing broadcasts in segments that have no use for them. Accuracy can be increased by increasing the size of the bloom filters, at the cost of increased area, latency and energy consumption. We determine the optimal design point in this regard empirically.

Smart page coloring: We explore using OS-based first-touch page coloring to further increase locality and reduce network energy. While improved locality will favorably impact any on-chip network, we believe that page coloring is even more compelling in our proposed network architecture for two reasons:

First, it has a positive influence on the accuracy of our filters, especially the out filters, and results in further reduction in energy. Figure 8 shows the accuracy rates when page coloring is enabled. We see that this improves to 97% on average from approximately 85% without page coloring. This can be explained by the fact that data is now more localized, and fewer lines are requested by cores outside the local segments, decreasing the amount of information stored by filters of the same size. We now see that only about 55% of messages leave the local bus and go out, compared to about 70% without page coloring. Second, since the granularity of energy consumption in the segmented bus is one ‘segment’, there are potentially more

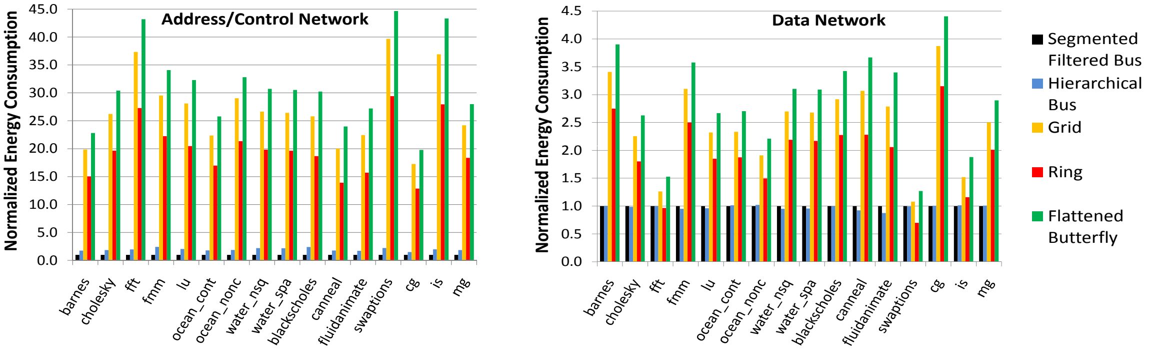


Figure 7. Energy consumption of various networks normalized to the segmented filtered bus

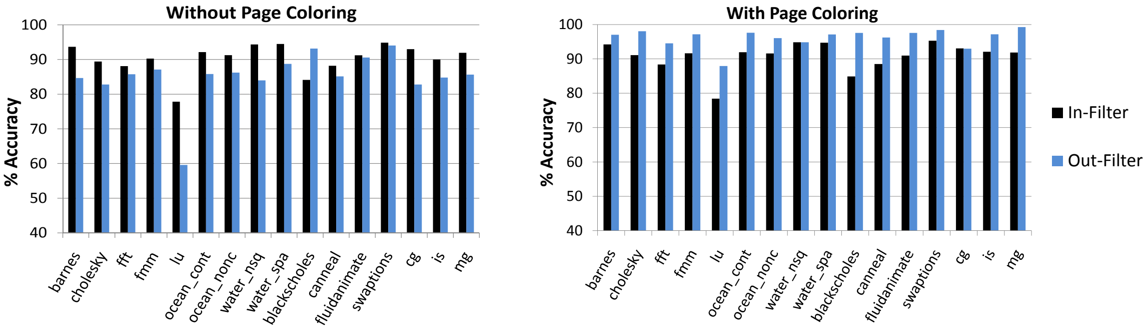


Figure 8. Accuracy rates of the bloom filters (note that there are no false negatives, only false positives)

opportunities to locate cache lines in slices that are accessible via a single sub-bus broadcast. In comparison, in packet switched networks, even if a line migrates to a core that is a few hops closer, there are still a few expensive router traversals before the transaction is complete.

As a representative comparison, Figure 9 shows the additional energy savings brought about by employing page coloring with the filtered bus network and the flattened butterfly network. We see an average of 28% additional energy savings than a non-page colored scheme with a best case improvement of 71% for the bus-based network. For the flattened butterfly, we see an average of 15% in additional savings with a best case of 44%. For the data network, we see an additional 26% benefit in the bus-based system compared to an additional 11% in a packet-switched network.

The variation in energy savings between different benchmarks can be attributed to variation in the percentage of accesses happening to private pages. For instance, 44% of all accesses in *mg* are to private pages and mapping these accurately through smart page coloring gives tremendous benefits. In *Swaptions*, on the other hand, less than 1% of accesses are to private pages, making page coloring less useful.

3.2.2 Performance Characteristics

It is clear that a bus-based network free from power-hungry routers is a sure win from an energy standpoint. It also turns out that if used effectively by stifling broadcasts, buses also perform as well as, if not better than, a packet-switched directory based system.

Execution time: Figure 10 shows the relative execution cycles over the entire parallel section of the benchmarks for various network organizations normalized to the filtered segmented bus. The "ideal" network is a zero latency network. It is seen that our proposed filtered segmented outperforms all the other networks. We see an average of 1% improvement compared to the best performing flattened butterfly network, with up to 6% improvement in execution time. This is primarily due to two factors: (a) inherent indirection in a directory based system, with multiple messages getting sent around for every transaction and (b) deep pipelines in complex routers increasing the no-load latency of the network.

This can be understood further by inspecting the latencies in both networks. From our experiments we see that the filtered segmented bus has an average latency of 16.4 cycles/transaction with the worst benchmark experiencing a delay of 18.3 cycles/transaction. This consists of the following major components: the part of the arbiter request/grant latency that cannot be fully overlapped with existing bus transactions (3.7 cycles), contention (about 1 cycle), bloom filter access (1.2 cycles) and link latency (10.5 cycles). Note that this is only the network latency and does not include the actual L2 cache lookup time. In the flattened butterfly network, there are on average 1.5 hops/message. Even with a bare minimum of two messages per transaction (due to home directory indirection), that equals 3 hops per transaction, which take 6 cycles of link latency and 9 cycles of router latency, giving a total of 15 cycles per transaction, even in the absence of contention. Very often, more than two messages are required per transaction, leading to larger network latencies.

While the performance advantage by itself is not largely significant, we believe that achieving a huge energy reduction with no performance loss is certainly an attractive de-

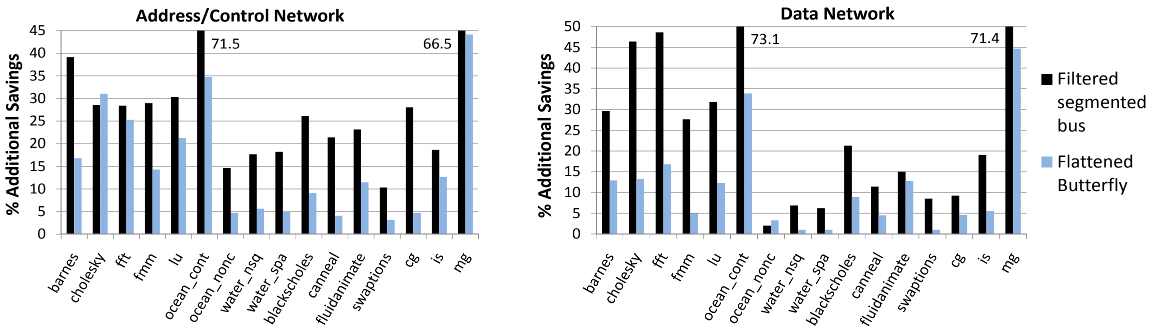


Figure 9. Additional savings achieved by applying smart page coloring to bus-based and packet-switched networks

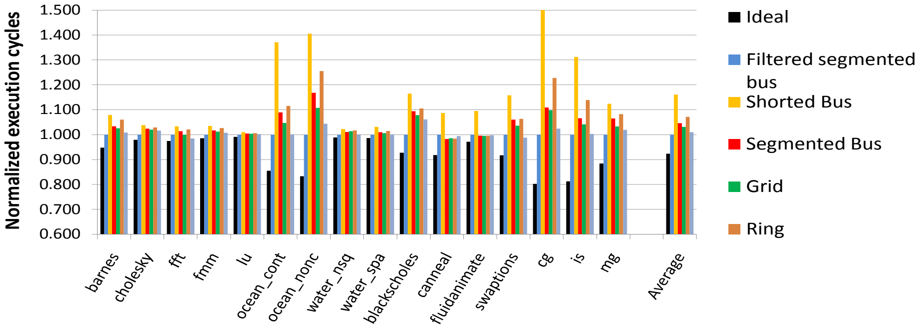


Figure 10. Relative execution time (cycles) for the various networks

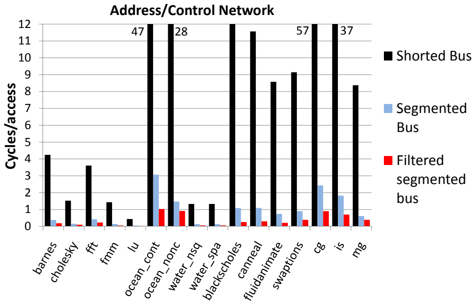


Figure 11. Contention in address network

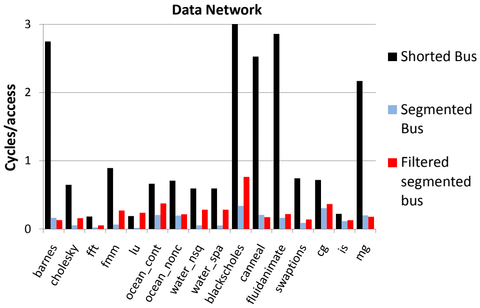


Figure 12. Contention in data network

sign point.

Contention: Contention is typically considered a major issue with shared media architectures like the bus. Figures 11 and 12 shows the average contention stall cycles per access for the bus-based models under consideration. It is clear that having a single shorted bus for all 16 cores and serializing all accesses results in massive contention of many cycles per access. By moving to the segmented bus, we pipeline every access into three stages and cores do not have to wait as long to get access to the bus. We see a significant drop in contention on moving to this design. We report the number of cycles that any core has to wait to get access to *all five* buses i.e., the contention per broadcast. Adding filters to this design further reduces the number of accesses on every bus and we see that contention drops even further. This is yet another advantage to stifling broadcasts in addition to saving energy. The filters thus contribute significantly to improving the scalability of the bus. Since a variable number of buses get activated on every transaction, we measure contention for each bus separately and report the maximum of these for every benchmark. In the worst case then, the net contention will be *less* than five times this number, which is still within acceptable limits.

Contention can be related to the rate at which messages are injected into the network. In our benchmarks, we see that the median gap between consecutive L1 misses is about 24 cycles (from any core), which is about once every 100 cycles on a per segment basis. Even the worst case benchmark has an average 10 cycle gap overall, with no more than one request per 40 cycles in a given segment. This benchmark (cg) sees the maximum contention, an average of 1 cycle per access. We thus see that the bus is never really overwhelmed by requests given that average transaction latency is 16.4 cycles.

3.2.3 Potential for router optimizations

From the discussion so far, it is clear that the main bottleneck to energy-efficiency of packet-switched networks are the routers. It is clear that if packet-switched networks are to become feasible in the future with large core-counts,

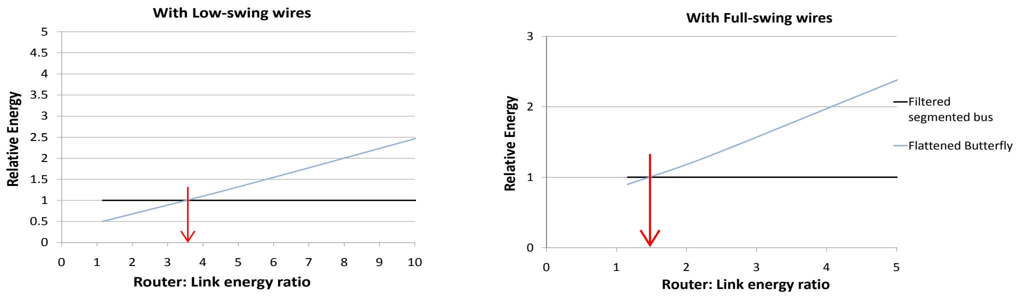


Figure 13. Energy consumption as a function of router:link energy ratio

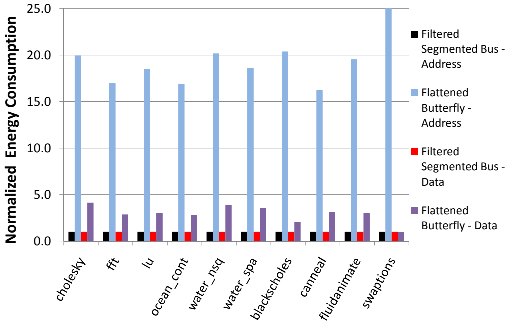


Figure 14. Energy characteristics at 32 cores

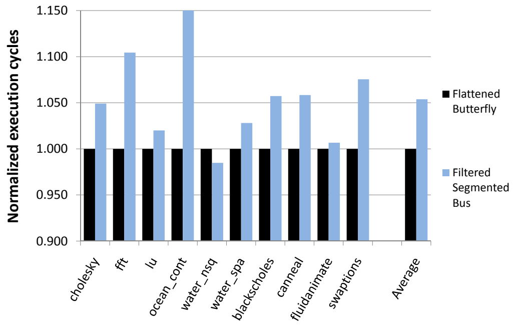


Figure 15. Performance characteristics at 32 cores

routers will necessarily have to be energy-optimized. Figure 13 shows the energy consumption of the address network for the filtered segmented bus and the flattened butterfly network as a function of the Router:Link energy ratio.

We see that for a packet-switched network to be more energy efficient than a bus, this ratio would have to be of the order of 3.5:1 if low-swing links are used and about 1.5:1 for full swing links. Our current numbers indicate this ratio to be of the order of 70-100 for low-swing and about 7-10 for full-swing, indicating that router optimization has a huge hill to climb in terms of energy-efficiency.

Data networks, on the other hand, have a single source and destination, both of which are known a priori, removing the need for indirection (as in directory protocols) or broadcast (as in snooping protocols). While the removal of routers still makes bus-based data networks an attractive low-energy choice with low-swing links, it is often better to employ a packet switched network if full-swing links are required to be used.

3.2.4 Scaling Characteristics

32 core system: On scaling to 32 cores, we retain segments of 4 cores each and expand the central bus to now have 8 nodes. Due to the same reasons as for the 16-core system, we continue to see significant energy savings as shown in Figure 14. We see an average of 19X reduction in energy with a best case saving of 25X.

With requests now being spread around, however, we see the *Out-filters* being exercised more and their average accuracy comes down to about 80% (for no increase in size relative to the 16-core case). There are more nodes requesting the central bus with slightly less efficient filters, increasing contention. We see average contention now rise to about 7 cycles/access (4.5 cycles/access in the data network), resulting in a small drop in overall performance. Figure 15 shows the relative performance of the filtered segmented bus and the flattened butterfly. We see an average 5% drop in performance, with a worst case of about 15%. Note that if contention were added to the flattened butterfly network, the performance gap would reduce. Nonetheless, considering the tremendous energy savings, we believe this is still a worthwhile tradeoff.

64 core system: For a 64-core system, we assume an 8x8 structure of tiles. If we retain a segment size of 4, we would have 16 sub-buses connected to a single large central bus. The length of the central bus would significantly increase its latency and subsequently the contention to get ownership. We therefore increase the segment size to 8 and have a central bus connecting 8 segments.

Figure 16 shows the relative energy consumption of such a structure and a flattened butterfly network. We see an average 13X decrease in energy in the data network and a 2.5X decrease in the data network, with a best case reduction of 16X and 3.3X. This, however, comes at a significant performance drop due to increased contention. As seen in Figure 17, there is a 46% increase in execution time compared to the flattened butterfly. To address this issue, we implement multiple address interleaved buses as described in the previous section. Accesses and contention now get spread around and execution time drops back to within 12% of the flattened butterfly.

4. Related Work

Various proposals in the past have attempted to address the issue of interconnect network design. Dally and Towles proposed packet-switched on-chip networks as a means to structure the wiring on-chip and provide reduced latency and increased bandwidth [16, 17]. Further research into this area has yielded numerous interesting ideas and proposals including the mesh [42, 45, 48], popular due to its relative simplicity and regularity, the flat-

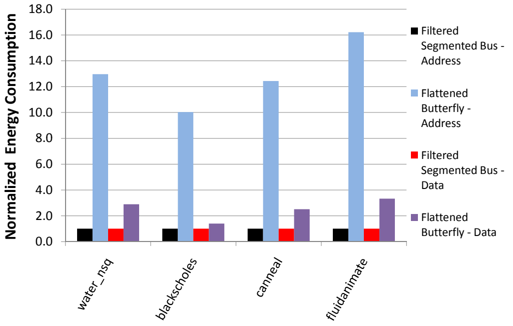


Figure 16. **Energy characteristics at 64 cores**

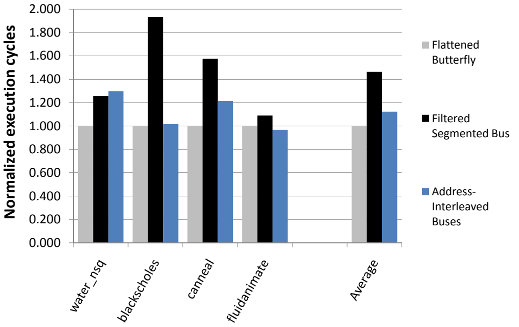


Figure 17. **Performance characteristics at 64 cores**

tened butterfly [25], using high-radix routers to reduce network diameter, MECS [20], using a one-to-many communication model enabling a high-degree of connectivity, and the ring [5], among others. Ainsworth et al. [2] study the interconnect architecture of the Cell processor using conventional latency and throughput characterization methods. Bourduas et al. [9] explore using a hierarchy of submeshes connected by a global ring interconnect in an effort to decrease the average hop count for global traffic. Many of these studies focus primarily on latency and bandwidth characteristics of on-chip networks. Wang et al. [47], however, characterize the impact of process technologies on network energies and use an analytical model to predict the most energy-efficient topology based on network size and architecture parameters. Many papers [27, 39, 46] have attempted router innovations to reduce the overheads of such packet-switched networks. Dally [18] proposed express cubes to provide a short path for non-local messages. Kumar et al. [28] attempt to approximate an ideal network by dynamically bypassing some intermediate routers with the use of express virtual channels. Martin et al. [33] propose a token coherence system to separate correctness from performance and simplify protocol verification. Kumar et al. [29] describe the design of a shared bus fabric and advocate co-design of interconnects and caches to improve overall performance.

IBM uses a snoop filter in its Blue Gene/P systems [41] based on the Jetty snoop filtering system [35]. A similar snoop filtering system is used by Strauss et al. [44] in a logical ring network to predict when requests have to actually snoop or be forwarded. Both these proposals are targeted at reducing the number of snoop lookups to reduce energy consumption in the tag arrays, whereas we propose filtering broadcasts to reduce network energy. Chinthamani et al. [12] study the design trade-offs for snoop filters in a dual-bus web server system to reduce traffic on the front-side-bus and reduce the average memory read latency.

The advantages of the broadcast bus have also been highlighted by recent work. Manevich et al. [31] advocate using a bus for specific purposes like DNUCA search and control signal broadcasts across the chip. Das et al. [19] propose a hierarchical network with local buses connecting small sets of cores, connected in a grid. We believe that while this is indeed a good direction to go in, the life of the bus can certainly be extended further, and that having many expensive routers on chip is not a feasible option due to power concerns. That paper too, considered page coloring, but did not consider several other innovations (segmented buses, broadcast stifling, low-swing links etc.) that are part of this paper.

5. Conclusions

It is clear that moving forward, power consumption will be a first order design constraint for processor architectures. In view of this, we propose extension of the life of bus-based networks beyond current projections to eliminate power hungry routers. In addition, retaining a snooping based protocol removes the need for indirection, decreases design complexity and reduces verification effort. We propose a segmented bus architecture with the use of bloom filters to stifle broadcasts as soon as possible, reducing energy consumption. Page coloring effectively increases locality and the accuracy of the bloom filters to further reduce energy consumption. Low-swing wires help mitigate the energy overheads of long bus links while multiple address-interleaved buses help mitigate their contention overhead. Overall, we see energy reductions of up to 31X with respect to the best packet switched network with no loss in performance.

References

[1] Virtutech Simics Full System Simulator. http://www.virtutech.com.

[2] T. W. Ainsworth and T. M. Pinkston. Characterizing the Cell EIB On-Chip Network. *IEEE Micro*, September/October 2007.

[3] M. Awasthi, K. Sudan, R. Balasubramonian, and J. Carter. Dynamic Hardware-Assisted Software-Controlled Page Placement to Manage Capacity Allocation and Sharing within Large Caches. In *Proceedings of HPCA*, 2009.

[4] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, D. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks. *International Journal of Supercomputer Applications*, 5(3):63–73, Fall 1991.

[5] L. Barroso and M. Dubois. Performance Evaluation of the Slotted Ring Multiprocessor. *IEEE Transactions on Computers*, vol.44(7), July 1995.

[6] C. Benia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. Technical report, Department of Computer Science, Princeton University, 2008.

[7] S. Borkar. Networks for Multi-Core Chips – A Contrarian View, ISLPED Keynote, 2007. www.islped.org/X2007/BorkarISLPED07.pdf.

[8] S. Borkar. Networks for Multi-Core Chips – A Controversial View. In *Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems (OCIN)*, 2006.

[9] S. Bourduas and Z. Zilic. A Hybrid Ring/Mesh Interconnect for Network-on-Chip Using Hierarchical Rings for Global Routing. In *Proceedings of NOCS*, 2007.

[10] M. Chaudhuri. PageNUCA: Selected Policies For Page-Grain Locality Management In Large Shared Chip-Multiprocessor Caches. In *Proceedings of HPCA*, 2009.

[11] C. O. Chen, N. Agarwal, T. Krishna, K. Koo, L. Peh, and K. C. Saraswat. Comparison of Physical Express Topologies and Virtual Express Topologies for Future Many-core On-Chip Networks. In *Proceedings of CMP-MSI Workshop*, 2009.

[12] S. Chinthamani and R. Iyer. Design and Evaluation of Snoop Filters for Web Servers. In *Proceedings of SPECTS*, 2004.

[13] S. Cho and L. Jin. Managing Distributed, Shared L2 Caches through OS-Level Page Allocation. In *Proceedings of MICRO*, 2006.

[14] D. E. Culler and J. P. Singh. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers, 1999.

[15] W. Dally. Report from Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems (OCIN), 2006. `http://www.ece.ucdavis.edu/~ocin06/`.

[16] W. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proceedings of DAC*, 2001.

[17] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 1st edition, 2003.

[18] W. J. Dally. Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks. *IEEE Transactions on Computers*, 40(9), 1991.

[19] R. Das, S. Eachempati, A. K. Mishra, N. Vijaykrishnan, and C. R. Das. Design and Evaluation of Hierarchical On-Chip Network Topologies for Next Generation CMPs. In *Proceedings of HPCA*, 2009.

[20] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. Express Cube Topologies for On-Chip Interconnects. In *Proceedings of HPCA*, 2009.

[21] R. Ho. *On-Chip Wires: Scaling and Efficiency*. PhD thesis, Stanford University, August 2003.

[22] R. Ho, T. Ono, F. Liu, R. Hopkins, A. Chow, J. Schauer, and R. Drost. High-speed and low-energy capacitively-driven on-chip wires. In *Proceedings of ISSCC*, 2007.

[23] A. Kahng, B. Li, L.-S. Peh, and K. Samadi. ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration. In *Proceedings of DATE*, 2009.

[24] C. Kim, D. Burger, and S. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Dominated On-Chip Caches. In *Proceedings of ASPLOS*, 2002.

[25] J. Kim, J. Balfour, and W. J. Dally. Flattened Butterfly Topology for On-Chip Networks. In *Proceedings of MICRO*, 2007.

[26] J. Kim, W. Dally, B. Towles, and A. Gupta. Microarchitecture of a High-Radix Router. In *Proceedings of ISCA*, 2005.

[27] J. Kim, C. Nicopoulos, D. Park, V. Narayanan, M. Yousif, and C. Das. A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks. In *Proceedings of ISCA*, 2006.

[28] A. Kumar, L. Peh, P. Kundu, and N. Jha. Express Virtual Channels: Towards the Ideal Interconnection Fabric. In *Proceedings of ISCA*, 2007.

[29] R. Kumar, V. Zyuban, and D. Tullsen. Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads, and Scaling. In *Proceedings of ISCA*, 2005.

[30] P. Kundu. On-Die Interconnects for Next Generation CMPs. In *Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems (OCIN)*, 2006.

[31] R. Manevich, I. Walter, I. Cidon, and A. Kolodny. Best of Both Worlds: A Bus-Enhanced NoC (BENoC). In *Proceedings of NOCS*, 2009.

[32] R. Marculescu, U. Ogras, L. Peh, N. Jerger, and Y. Hoskote. Outstanding Research Problems in NoC Design: System, Microarchitecture and Circuit Perspectives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(1), January 2009.

[33] M. Martin, M. Hill, and D. Wood. Token Coherence: Decoupling Performance and Correctness. In *Proceedings of ISCA*, pages 182–193, June 2003.

[34] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga. Prediction Router: Yet Another Low Latency On-Chip Router Architecture. In *Proceedings of HPCA*, 2009.

[35] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary. JETTY: Filtering Snoops for Reduced Energy Consumption in SMP Servers. In *Proceedings of HPCA*, 2001.

[36] S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb. The Alpha 21364 Network Architecture. In *IEEE Micro*, volume 22, 2002.

[37] R. Mullins, A. West, and S. Moore. Low-Latency Virtual-Channel Routers for On-Chip Networks. In *Proceedings of ISCA*, 2004.

[38] N. Muralimanohar and R. Balasubramonian. Interconnect Design Considerations for Large NUCA Caches. In *Proceedings of ISCA*, 2007.

[39] C. Nicopoulos, D. Park, J. Kim, V. Narayanan, M. Yousif, and C. Das. ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers. In *Proceedings of MICRO*, 2006.

[40] L.-S. Peh and W. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. In *Proceedings of HPCA*, 2001.

[41] V. Salapura, M. Blumrich, and A. Gara. Design and Implementation of the Blue Gene/P Snoop Filter. In *Proceedings of HPCA*, 2008.

[42] K. Sankaralingam, R. Nagarajan, P. Gratz, R. Desikan, D. Gulati, H. Hanson, C. Kim, H. Liu, N. Ranganathan, S. Sethumadhavan, S. Sharif, P. Shivakumar, W. Yoder, R. McDonald, S. Keckler, and D. Burger. The Distributed Microarchitecture of the TRIPS Prototype Processor. In *Proceedings of MICRO*, 2006.

[43] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: A Many-Core x86 Architecture for Visual Computing. *ACM Transactions on Graphics*, 27(3), August 2008.

[44] K. Strauss, X. Shen, and J. Torrellas. Flexible Snooping: Adaptive Forwarding and Filtering of Snoops in Embedded-Ring Multiprocessors. In *Proceedings of ISCA*, 2006.

[45] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar. An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. In *Proceedings of ISSCC*, 2007.

[46] H.-S. Wang, L.-S. Peh, and S. Malik. Power-Driven Design of Router Microarchitectures in On-Chip Networks. In *Proceedings of MICRO*, 2003.

[47] H.-S. Wang, L.-S. Peh, and S. Malik. A Technology-aware and Energy-oriented Topology Exploration for On-chip Networks. In *Proceedings of DATE*, 2005.

[48] D. Wentzlaff, P. Griffin, H. Hoffman, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal. On-Chip Interconnection Architecture of the Tile Processor. In *IEEE Micro*, volume 22, 2007.

[49] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of ISCA*, 1995.