Sustainable Computing: Informatics and Systems 3 (2013) 36–46

Contents lists available at SciVerse ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

High-throughput, energy-efficient network-on-chip-based hardware accelerators

Turbo Majumder, Partha Pratim Pande*, Ananth Kalyanaraman

School of Electrical Engineering and Computer Science, Washington State University, Pullman, United States

ARTICLE INFO

Article history:
Received 25 October 2012
Accepted 28 January 2013

Keywords:
Network-on-chip
Hardware accelerator
Energy efficient
Long-range links
Wireless

ABSTRACT

Several emerging application domains in scientific computing demand high computation throughputs to achieve terascale or higher performance. Dedicated centers hosting scientific computing tools on a few high-end servers could rely on hardware accelerator co-processors that contain multiple lightweight custom cores interconnected through an on-chip network. With increasing workloads, these many-core platforms need to deliver high overall computation throughput while also being energy-efficient. Conventional multicore architectures can achieve a limited computational throughput due to the inherent multi-hop nature of the on-chip network infrastructure. By inserting long-range links that act as shortcuts in a regular network-on-chip (NoC) architecture, both the achievable bandwidth and energy efficiency of a multicore platform can be significantly enhanced. In this paper, we first propose a NoC-driven use-case model for throughput-oriented scientific applications, and subsequently use the model to study the effect of using long-range links in conjunction with different resource allocation strategies on reducing the overall on-chip communication and enhancing computational throughput. NoCs with both wired and on-chip wireless links are explored in the study. We also evaluate our NoC-based platforms with respect to energy-efficiency and power consumption. We analyze how throughput and power consumption are correlated with the statistical properties of the application traffic. In addition, we compare and analyze chip-level thermal profiles for these alternatives. Our experiments using kernels from a popular phylogenetic inference application suite show that we can deliver computation throughput over 10^{11} operations per second, consuming ~0.5 nJ per operation, while ensuring that on-chip temperature variation is within 26 °C.

1. Introduction

Many scientific computing disciplines have seen a significant increase in the availability of parallel algorithms and high performance computing (HPC) tools owing to high run-time complexities and/or the data-intensive nature underlying the computation. This is particularly true with emerging application domains, such as bioinformatics and computational biology, in addition to more traditional applications requiring HPC resources. In all such cases, new data-generation technologies are placing an enormous stress on software tools to perform beyond terascale to peta- and exascale. Given the diversity of tools and the need to cater to a wide user-base, it is becoming common practice, even within academic settings, to have a dedicated center that hosts a variety of scientific computing tools on a few high-end data servers. The throughput requirement that these multi-user servers need to meet can be substantial since the servers can be expected to service concurrent requests from a variety of applications, each with differing resource requirements. These servers would often consist of multicore hardware accelerator co-processor platforms where the cores are designed to accelerate targeted operations and are interconnected with an on-chip network. A similar setup is also becoming common practice, albeit on a larger scale, in cloud solution providers (e.g. [1]). While throughput is important, it is necessary to restrain energy consumption and power dissipation in these hardware accelerators. Multicore platforms based on network-on-chip (NoC) are known to have inherent advantages with respect to both throughput and energy-efficiency [2]. In addition, various studies have shown the efficacy of NoC driven platforms to accelerate specific target applications (e.g. [3]).

In this paper, we propose, design and evaluate NoC-based platforms for enhancing the computation throughput of scientific applications. Our evaluation of the NoC-based platforms includes analysis of the resulting performance, energy-efficiency and power consumption, and thermal profiling. More specifically, we analyze how throughput and power consumption are correlated with the statistical properties of the application traffic. In addition, we compare and analyze chip-level thermal profiles to identify hotspot distribution and correlate them with architecture-level design choices.

* Corresponding author. Tel.: +1 509 335 5223.
E-mail addresses: tmajumde@eecs.wsu.edu (T. Majumder), pande@eecs.wsu.edu (P.P. Pande), ananth@eecs.wsu.edu (A. Kalyanaraman).

2210-5379/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.suscom.2013.01.001

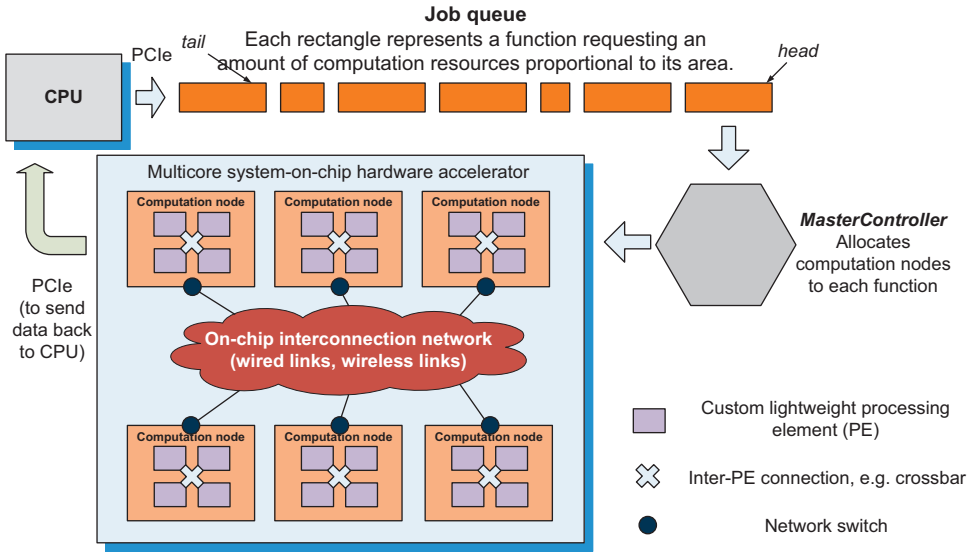

Fig. 1. Illustration of our NoC-based use-case model designed for hardware acceleration of throughput-oriented scientific applications.

In order to design our platform, we assume the following *use-case model* (see Fig. 1): a CPU runs the parent process and communicates via an interface (e.g. PCIe) to a multicore system-on-chip that acts as a hardware accelerator for specific computation-heavy kernels. There is a queue of jobs offloaded by the CPU to the hardware accelerator and an allocation unit (*MasterController*) assigns the requested computational resources from the hardware accelerator to the job at the head of the queue. Once some computation resources are assigned to a job, they stay busy till the execution of that particular job concludes, and the result is sent back to the CPU through a similar interface (e.g. PCIe). Each computational resource is a lightweight custom core embedded in a NoC. This work aims to characterize such hardware accelerator platforms in terms of overall computation throughput, energy consumption, power dissipation and thermal profiling.

The computational footprint of many scientific applications fits the proposed use-case model, for example, the use of servers that host application programs to implement standard functions in phylogenetic inference [6], genome/gene sequencing [7], climate modeling and weather prediction [8], etc. For instance, a typical genome assembly algorithm farms out billions of pairwise sequence alignment tasks, each of which aligns two strings of small lengths (e.g. 100–500 base pairs) and can use a small number of cores (e.g. 8–16) [9]. As another example, consider the problem of computing phylogenetic inference using maximum likelihood (ML) [10], where one typically needs to carry out billions of independent tree evaluations, each of which internally performs a small number of floating point calculations using a few cores. In such applications, enhancing overall throughput in computation translates to shorter time to solution. To this end, integration of many cores using an on-chip network (or NoC) presents an attractive model of computation, not only due to the availability of a large number of cores, but also because the computation within the individual tasks in many of these applications (e.g. sequence alignment in the genome assembly problem [16]) can be designed to take advantage of fine-grain on-chip parallelism involving a fixed number of cores.

For any NoC design, the choice of the on-chip network architecture is an important determinant of the overall throughput and energy efficiency achieved by the many-core system. Introduction of long-range links in regular architectures such as a mesh reduces the overall network diameter and enhances throughput by reducing inter-core communication latency [4]. It has been shown that the use of on-chip wireless links to implement these shortcuts leads to significant savings in latency and energy, even considering the overhead of wireless transceivers [5]. Consequently, in this paper, we design and evaluate NoC-based platforms consisting of long-range on-chip wireless links in addition to standard wired links in a particular network topology. To the best of our knowledge, the study reported in this paper represents the first attempt at designing and empirically characterizing NoC-driven accelerator platforms built using both wired and wireless links for throughput-oriented scientific applications.

2. Related work

Use of hardware accelerators for scientific computation is the subject of substantial ongoing research. A variety of platforms, including those based on FPGA, GPU, CBE, general-purpose multicores and custom multicores, have been considered in existing research for biocomputing applications such as sequence analysis, phylogenetic inference, molecular dynamics and molecular docking [11]. Hardware accelerators using FPGA have been developed for implementing ClustalW [12], which is a popular program for carrying out multiple sequence alignment of genomes. The sequence search tool BLAST has been accelerated with the help of Cell Broadband Engine (CBE), consisting of a 64-bit Power Processor Element (PPE) and eight Synergistic Processing Elements (SPEs) [13], Annapolis Microsystems Wildstar II-Pro board with two Xilinx Virtex-II FPGAs [14], and nVidia GeForce 7800 GTX GPU [15]. However, the best performance in PSA is reported by a NoC-based implementation [16] owing to its custom core architecture and interconnection topology. In phylogenetic inference, prior work has explored HW/SW co-design using FPGA-based hardware accelerator [17], FPGA platforms with DSP slices to speed up floating-point operations [18], using an FPGA platform to implement a co-processor for whole genome Maximum Parsimony (MP) phylogeny reconstruction [19], optimizing a version of a Maximum Likelihood (ML) phylogeny reconstruction for CBE [20], and comparison of CBE, GPU and general-purpose multiprocessor for speeding up a Bayesian phylogenetic inference program [21]. A comparison of the acceleration achieved across these studies and particularly the comparative study in [21] show that multicores have the best overall throughput performance and the other platforms suffer from inter-processor communication bottlenecks. Consequently, network-on-chip (NoC) based multicore hardware accelerator

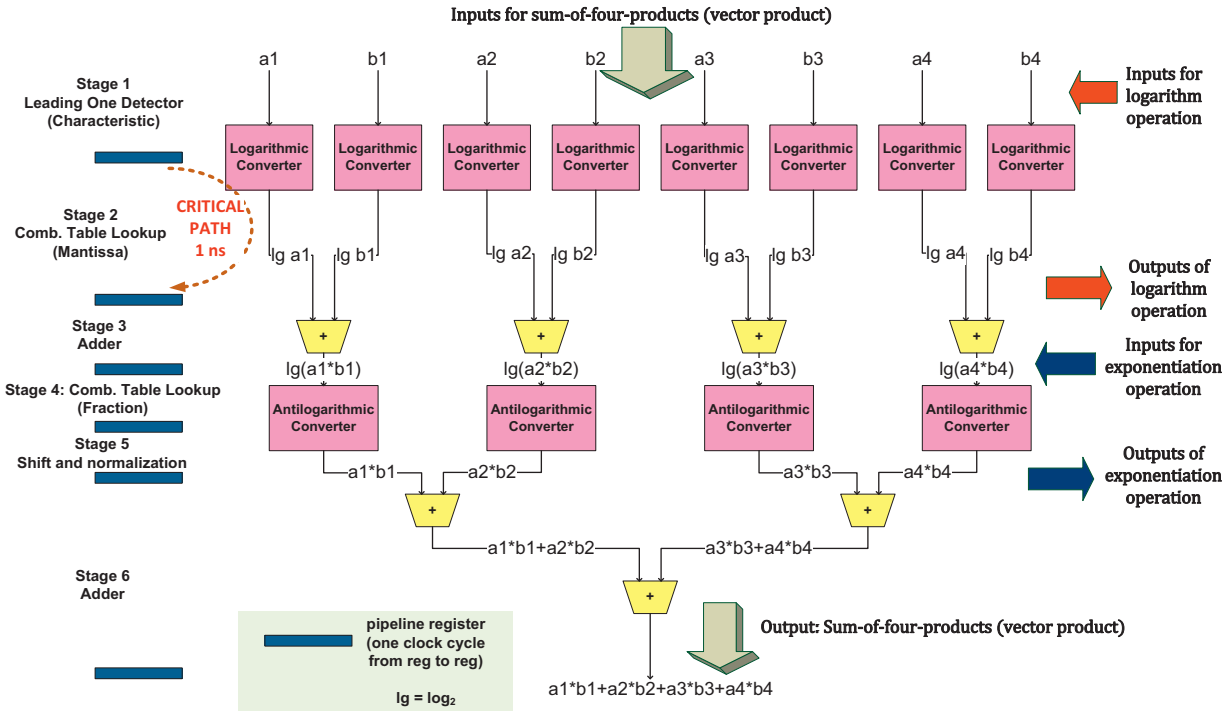

Fig. 2. Datapath architecture of each computation core in a PE to calculate log, exp and vector products.

platforms have been shown to deliver superior performance for MP [22] and ML [23] phylogeny reconstruction.

Among other compute-intensive domains, hardware acceleration is of critical importance in real-time image and video processing, as evident in the substantial research being carried out in the area, for example [24]. An analytical optimization framework for a multi-accelerator SoC, given a sequential workload and a set of potential accelerators, is described in [25]. Most of such accelerators are limited in the number of cores they can integrate in order to guarantee efficient inter-core communication for delivering high throughput. Although standard NoC-based platforms have an edge over others, multi-hop communication often proves to be the principal bottleneck in achieving high throughput.

Insertion of long range wired links following principles of small-world graphs [4] have shown to enhance NoC performance. However, this scheme implements the long-range links with conventional wires. It has been already shown that beyond a certain length, wireless links are more energy efficient than conventional metal wires. Hence, the performance improvement by using long-range wireless links will be more than that using wired links. Designs of small-world based hierarchical wireless NoC architectures introduced and elaborated in [5] demonstrate this. We leverage the benefits of using long-range wireless shortcuts on a NoC to achieve high-throughput computation for applications, while maintaining low energy consumption and power dissipation. We also show how different models of usage of wireless long-range links present a throughput-power tradeoff.

3. NoC design with long-range links

We present the design of a multicore system-on-chip, where the cores consist of lightweight custom-designed processing elements (PEs), and the on-chip network is a folded torus (network choice explained in Section 3.2). We insert long-range shortcuts using on-chip wireless links on top of the folded torus, and explore different strategies to allocate the computational resources of the system to the application. The details of the system design, wireless shortcut placement, resource allocation and routing are described in this section.

3.1. Processing element (PE)

Scientific applications need to carry out a wide range of operations (e.g. logarithm, exponentiation, multiplication, integer comparison, trigonometric functions, etc.). For the sake of design and evaluation in this paper and without loss of generality, we built a PE that performs only a subset of these calculations (using phylogenetic inference as a demonstration study). It is to be noted that the PE design can be modified to accommodate other kinds of granular calculations without changing the overall system design and evaluation methodology. We designed a homogeneous many-core system, where a PE computes vector products on floating point numbers and elementary functions like logarithms and exponentials. A PE is one computation node and communicates with other PEs (computation nodes) through the respective network switches and the on-chip network. The number of such nodes represents the system size, N, of the many-core system.

In order to include fine-grain parallelism in our study, we designed each PE to consist of four similar computation cores. The function of the core is to carry out arithmetic operations at the heart of hardware acceleration. We use fixed-point hybrid number system (FXP-HNS) [26], an efficient and accurate number system to represent floating point numbers. We use 64-bits for number representation; as such our core datapath is 64-bit wide. The architecture of the datapath of a computation core within each PE is shown in Fig. 2. The datapath consists of six pipeline stages, with the functions of each stage indicated in the figure. We consider logarithm and exponentiation as *basic operations* that the PE is designed to accelerate, while vector product (sum-of-four-products) is a *compound operation* involving the basic operations. Logarithm and exponential operations (see *logarithmic converter* and *antilogarithmic converter* in Fig. 2) are based on piecewise-linear table-based approximations described in [26], and implemented with logic circuits. In addition, each PE has 0.5 MB memory in the form of register

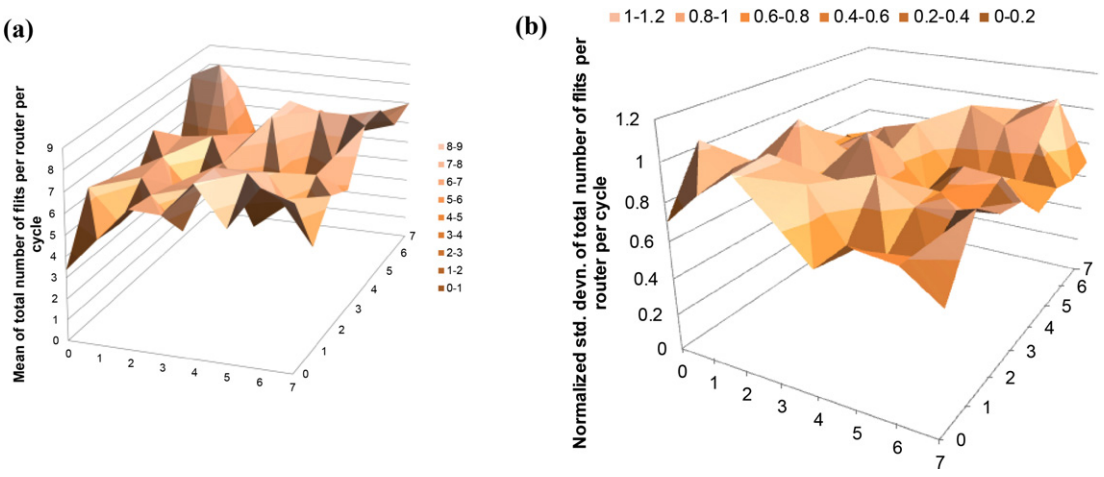

Fig. 3. The number of flits per cycle ((a) mean, and (b) normalized standard deviation) within routers of an 8×8 folded torus network. The horizontal dimensions denote the processor coordinates of the torus, while the vertical dimension denotes the observed flits per cycle (mean and standard deviation). The absence of distinctive peaks in temporal statistics indicates the higher suitability of fully-distributed, regular interconnection topologies such as a mesh or torus, as opposed to linear or hierarchical topologies such as bus or trees.

banks to store inputs and computation outputs. We used Verilog HDL to design the PE along with a wrapper for instruction decoding, data fetching and data write-back. We synthesized the design with 65 nm standard cell libraries from CMP [27]. We determined the clock frequency of 1 GHz based on the critical path (Stage 2) shown in Fig. 2.

3.2. Network architecture

The choice of the underlying interconnection fabric topology is determined from the perspective of the application and VLSI implementation. Our target is the class of applications that spawn a stream of independent jobs (constituent functions) that individually require variable amounts of computation resources. Communication occurs only among nodes catering to a single job during its execution. The location of these nodes on the network is a variable for every instance of an allocated job, leading to arbitrary point-to-point communication. The traffic patterns are hence dynamically changing and steady-state characteristics do not indicate any clustering or traffic localization, as shown in a sample statistical analysis of the traffic pattern over time in Fig. 3. Distributed network architectures are generally better suited for such traffic patterns. Consequently, we use a *folded torus*. From the VLSI implementation perspective, a torus is a scalable network architecture whose regularity provides for easier timing closure and reduces dependence on interconnect scalability [28]. All inter-node links in the folded torus are one-hop links with respect to the 1 GHz clock used. As mentioned above this clock frequency requirement arises from the critical path constraint in the PE. As mentioned earlier, our datapath is 64-bit wide. We designate our flit size to be 64 bits and split each inter-node message into three flits – header, body and tail. As a result, each inter-node link has a minimum bandwidth of 64 Gbps.

3.3. Long-range on-chip wireless links

In Section 3.2, we described our underlying network topology that suits a distributed traffic pattern. However, we would ideally want the average internode distances between nodes catering to the same job to be as low as possible, or in other words, we would want the nodes catering to a particular job to be all contiguous to one another. This cannot be guaranteed in practice because different jobs needing different number of resources could get submitted in real-time (as we further explain in Section 3.4). This could force any allocation method to either wait for all required nodes to be available contiguously (the effect of which could be a significant delay in execution time coupled with a non-optimal use of the cores) or map the job on nodes that could potentially be non-contiguous along the network (as elaborated in Section 3.4). In the latter approach, large physical separation of these nodes on the network could lead to a significant communication overhead. From the network architecture point of view, bridging these gaps is possible through the use of long-range point-to-point shortcuts. Introduction of shortcuts on regular architectures have been shown to provide significant improvements in latency and network throughput for different kinds of applications [4]. Implementing these shortcuts using metal wires inherits the issues associated with long wires, viz. transmission delay and large power dissipation. High transmission delay makes it impossible to guarantee one-hop transmission. Use of on-chip wireless shortcuts overcomes these drawbacks [5].

3.3.1. Physical layer

Suitable on-chip antennas are necessary to establish the wireless links. It has been shown that wireless NoCs designed using carbon nanotube (CNT) antennas can significantly outperform conventional wireline counterparts [5]. Antenna characteristics of CNTs in the THz frequency range have been investigated both theoretically and experimentally [29]. Such nanotube antennas are good candidates for establishing on-chip wireless communications links and are henceforth considered in this work. CNT antennas can be used to assign different frequency channels to pairs of communicating source and destination nodes. This enables creation of dedicated and non-overlapping channels using the concept of frequency division multiplexing. This is implemented by using CNTs of different lengths, which are multiples of the wavelengths corresponding to the respective carrier frequencies. With currently available technology, it is possible to create 24 non-overlapping wireless channels, each capable of sustaining a data rate of 10 Gbps using CNT antennas. Technology-specific details on CNT are discussed in [5]. We determine the number of wireless links in our system based on the bandwidth each link needs to support. As mentioned earlier, each wireless (inter-node) link needs to sustain a bandwidth of 64 Gbps. Since each wireless channel can provide a bandwidth of 10 Gbps, we need to combine 7 channels per link (delivering up to 70 Gbps bandwidth). Hence, the maximum

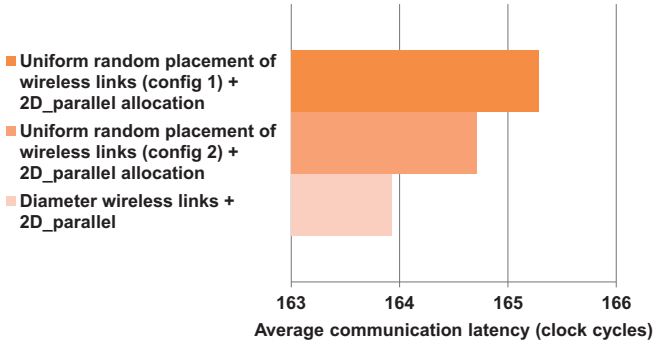

Fig. 4. Comparison of average network communication latencies for different wireless link placements. Even small increases in communication latency have shown to lead to significantly degrade performance.

number of single-hop wireless links we can implement is $\lfloor 24/7 \rfloor = 3$. Note that we could increase the number of wireless links providing the same bandwidth when future technology supports more than 24 non-overlapping channels.

3.3.2. Wireless link placement

Ideally, the placement of wireless links should be dictated by the demands in the traffic patterns generated by the target scientific application. For the kind of throughput-oriented scientific applications targeted in this paper, however, it is not possible to statically predict any particular traffic pattern because the underlying communication requirement could be arbitrary. The sets of communicating nodes executing a single task could be spread out over the whole network, thereby generating arbitrary point-to-point traffic over time (as corroborated by our observations made in Fig. 3). In fact, we observed the probability of non-local interaction between nodes is highest when the separation between them is equal to the diameter of the network. We have experimentally verified this observation by placing wireless links according to various considerations – based on uniform random traffic assumption, and along diameters of the network. As shown in Fig. 4, wireless link placement along the diameter leads to the lowest network latency among the possibilities considered. This observation can be explained by the fact that the most efficient node allocation method described in Section 3.4 divides the network into four quadrants, tries to allocate nodes locally, and the need for long-range links arises when allocated nodes are non-contiguous and lie in neighboring quadrants. It can be easily seen from Fig. 5 that the mean distance between adjacent quadrants is the network diameter of the folded torus. Since there are only 3 wireless links as explained earlier, we maximize their coverage by placing them along diameters of the folded torus with almost equal angular separation between each pair of diameters, as shown in Fig. 5. This ensures that nodes in all sections of the network have similar accessibility to a wireless link.

3.4. Dynamic node allocation

A network node is *busy* during the execution of a job by the PE; it is *available* otherwise. The computation nodes (PEs) continually send their busy/available status to the allocation unit, *MasterController* (see Fig. 1). When a job requests computation resources, *MasterController* allocates the requisite number of available computation nodes from the system. The nodes thus allocated form a *partition* during the course of function execution and communicate with one another. A desired feature of a partition is that its constituent nodes are co-located so as to minimize the average number of hops spent in message transfers. To this end, a good allocation

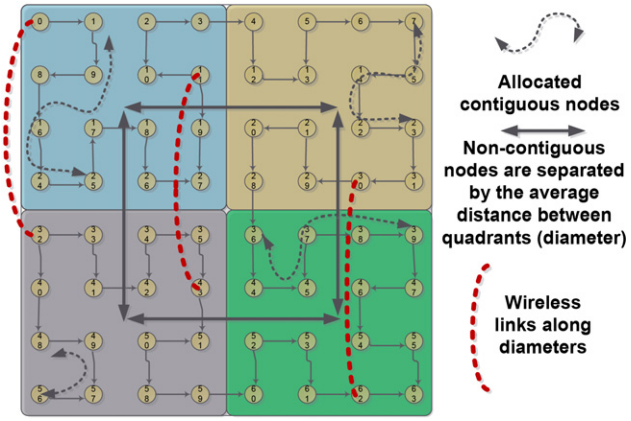

Fig. 5. Non-contiguous nodes and long-range communication requirements leading to wireless link placement along diameters. Although not shown, each node is connected through wired links to four of its neighboring nodes as dictated by the torus topology. Most of our allocation strategies consider the nodes along the Hilbert curve while also factoring the presence/absence of wireless hubs at intermediate nodes.

strategy should ensure co-locality without incurring a large allocation time overhead. Simple approaches like breadth-first search do not fit these criteria. We present the following allocation methods, which can be classified into *wireless-agnostic* and *wireless-aware* methods. We also make use of the locality-preserving, space-filling Hilbert curve [30] for allocation. The resultant allocated partitions are denoted *A-type* if all nodes belonging to that partition are contiguous along wired links on the folded torus; else the partition is *B-type*.

3.4.1. Parallel best-fit allocation using multiple Hilbert curves

This allocation strategy preferentially looks for a partition with contiguous nodes to maximize co-locality, and parallelizes the search in order to increase the probability of a hit. The algorithm is as follows:

1. First, we use four Hilbert curves on a square folded torus. These four curves are obtained by using right-angle rotation operations of a single Hilbert curve.
2. We further divide each of the four Hilbert curves into four *segments*, one from each quadrant – thereby resulting in a total of 16 segments (see Fig. 5). *MasterController* now has 16 heads, each of which is responsible for scanning a segment. All 16 heads act in parallel.
3. Each head now preferentially looks for an A-type partition in its segment. The first head to find such a partition returns it to the requesting job and interrupts all the other scanning heads.
4. In case no A-type partition is found after each head has finished scanning its segment, *MasterController* carries out a serial scan along a Hilbert curve and allocates available nodes as they are encountered.

This method of allocation is wireless-agnostic because we do not make use of the information regarding the location of wireless shortcuts. Systems using this method of allocation are denoted by *2D_parallel* if they do not use wireless shortcuts, and *2D_parallel + wireless* if wireless shortcuts are used only during message transfers.

3.4.2. Wireless-first allocation using Hilbert curve

This is a wireless-aware allocation method in which *MasterController* looks for available node pairs directly connected by a wireless shortcut. If such a pair is available, they are allocated to the requesting job. *MasterController* then serially scans for the remaining

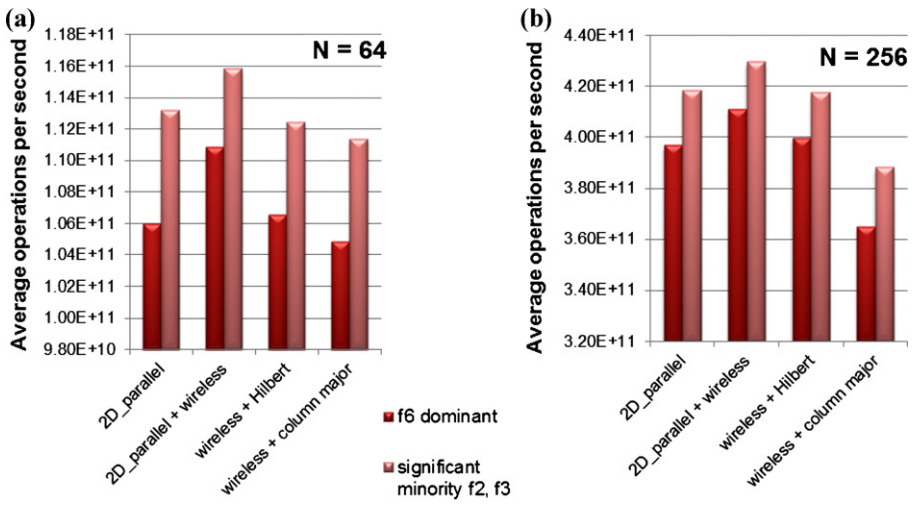

Fig. 6. Computation throughput across different network architectures, system sizes and job loads.

nodes following a Hilbert curve starting from a terminal node of the wireless shortcut. Since only nodes belonging to the same partition communicate with one another, this method ensures that wireless shortcuts are fully utilized. In case no wireless shortcut is available at the time of allocation, nodes are allocated based on a serial scan along the Hilbert curve. Systems using this allocation method are denoted by *wireless + Hilbert*.

3.4.3. Wireless-first, column-major allocation

This is another wireless-aware allocation method, which looks for available wireless shortcuts to be allocated first. The remaining nodes are allocated following the direction of wireless shortcuts such that the nodes in the partition are aligned with the shortcut, so as to maximize the traffic the shortcut carries. As shown in Fig. 5, the wireless shortcuts are placed along the vertical diameters (columns) of the folded torus. Hence, the node allocation also follows a *column-major* ordering. The major benefit of this method is that a wireless shortcut can potentially carry traffic from partitions that do not directly include it but are closely aligned with it. Systems using this allocation method are denoted by *wireless + column-major*.

3.5. On-chip routing

As mentioned earlier, we adopt wormhole routing to exchange three-flit messages among nodes of a partition. Network switches are based on the designs presented in [31]. The general routing policy follows dimension order routing, in this case XY routing. For *2D_parallel* systems that do not have wireless shortcuts, we use a restricted form of this routing policy. This policy applies only to A-type partitions and disallows paths outside the partition boundary.

For routing in the presence of wireless shortcuts, we need information about the wireless links closest to a source-destination pair, and the bandwidth provided by such links. This information is known beforehand and is available to the router. Based on this knowledge, the router chooses a path via a wireless shortcut if that entails fewer hops to transfer a message between a source-destination pair. The message follows deadlock-free

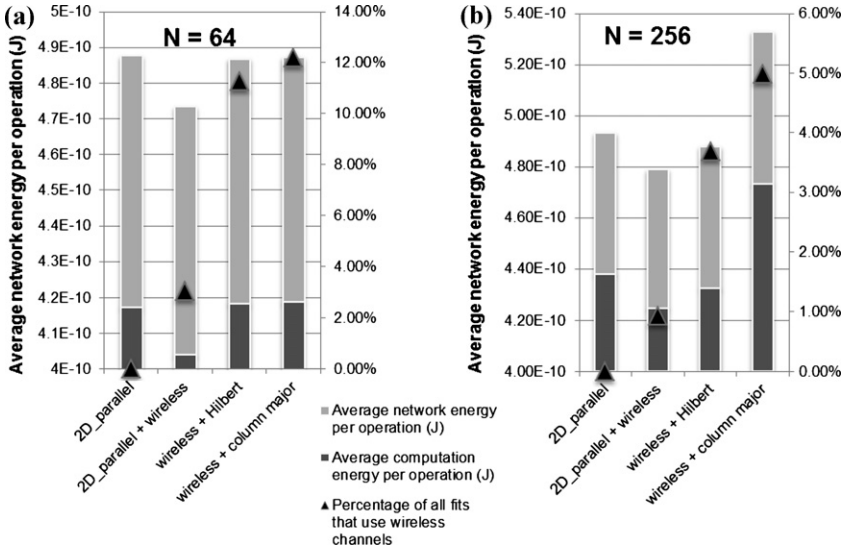

Fig. 7. Proportion of flits using wireless shortcuts and energy consumption across network architectures and system sizes.

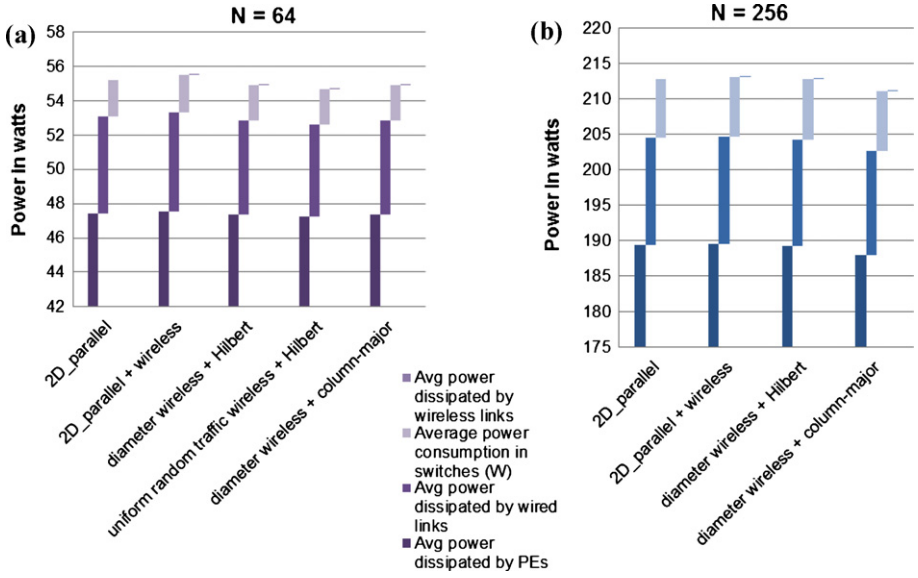

Fig. 8. Average power dissipation in different NoC architectures and system sizes.

south-last routing [4] when involving wireless shortcuts, and XY routing when following wired-only paths between a source and a destination.

4. Experimental results

4.1. Experimental setup

The computation core has a datapath width of 64 bits and provides a number representation accuracy of $\sim 10^{-15}$. As mentioned earlier, our PE integrates four such computation cores. We synthesized Verilog RTLs for the PEs, the network switches and *MasterController* with 65 nm standard cell libraries from CMP [27]. Our clock period of 1 ns comes from the critical path constraint in the core datapath as mentioned in Section 3.1 and shown in Fig. 2. We verified that our design meets all timing constraints, and evaluated power consumption. We laid out the wired NoC interconnects and determined their physical parameters (power dissipation, delay) using the extracted parasitics (resistances and capacitances). We verified that all wired links could be traversed within one clock cycle. Each wireless link consists of seven channels of 10 Gbps each, providing a total link bandwidth of 70 Gbps. For the wireless links, we considered an energy dissipation of 0.33 pJ/bit as reported in [5] to include the energy consumed in the transceiver circuitry and the antennas, and used these to evaluate the total energy consumption of our system. In order to carry out chip-level thermal analysis we used the data on power consumption so obtained with HotSpot 5.0, an accurate and fast thermal model suitable for use in architectural studies [32].

We experimented with the allocation methods mentioned in Section 3.4. We used system sizes of $N = 64$ and $N = 256$ in our experiments. We model the NoC-based multicore platform as a co-processor connected using a PCIe interface. We modeled a PCI Express 2.0 interface using Synopsys™ Designware™ IP PCI Express 2.0 PHY implemented on 65 nm process and operating at 5.0 Gbps. We use a 32-lane PCIe 2.0 for our simulation.

For experimental studies, we use function kernels from a Maximum Likelihood-based phylogenetic reconstruction software called RAxML version 7.0.4 [33,34]. A detailed profiling of RAxML runs using the GNU *gprof* utility reveals that a small set of functions consume a predominant portion (>85%) of the runtime. These functions are offloaded to our NoC-based accelerator co-processor and

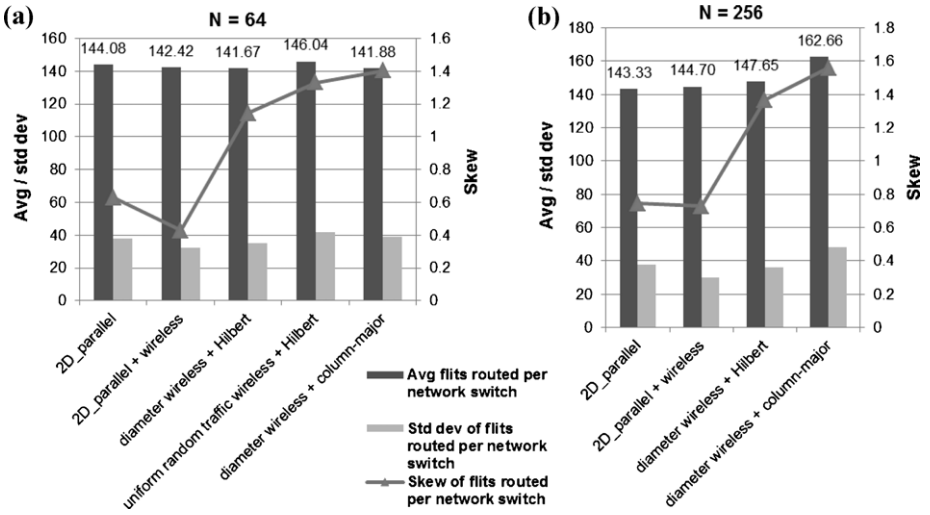

Fig. 9. Average, standard deviation and skew of flits routed per network switch across NoC architectures and system sizes.

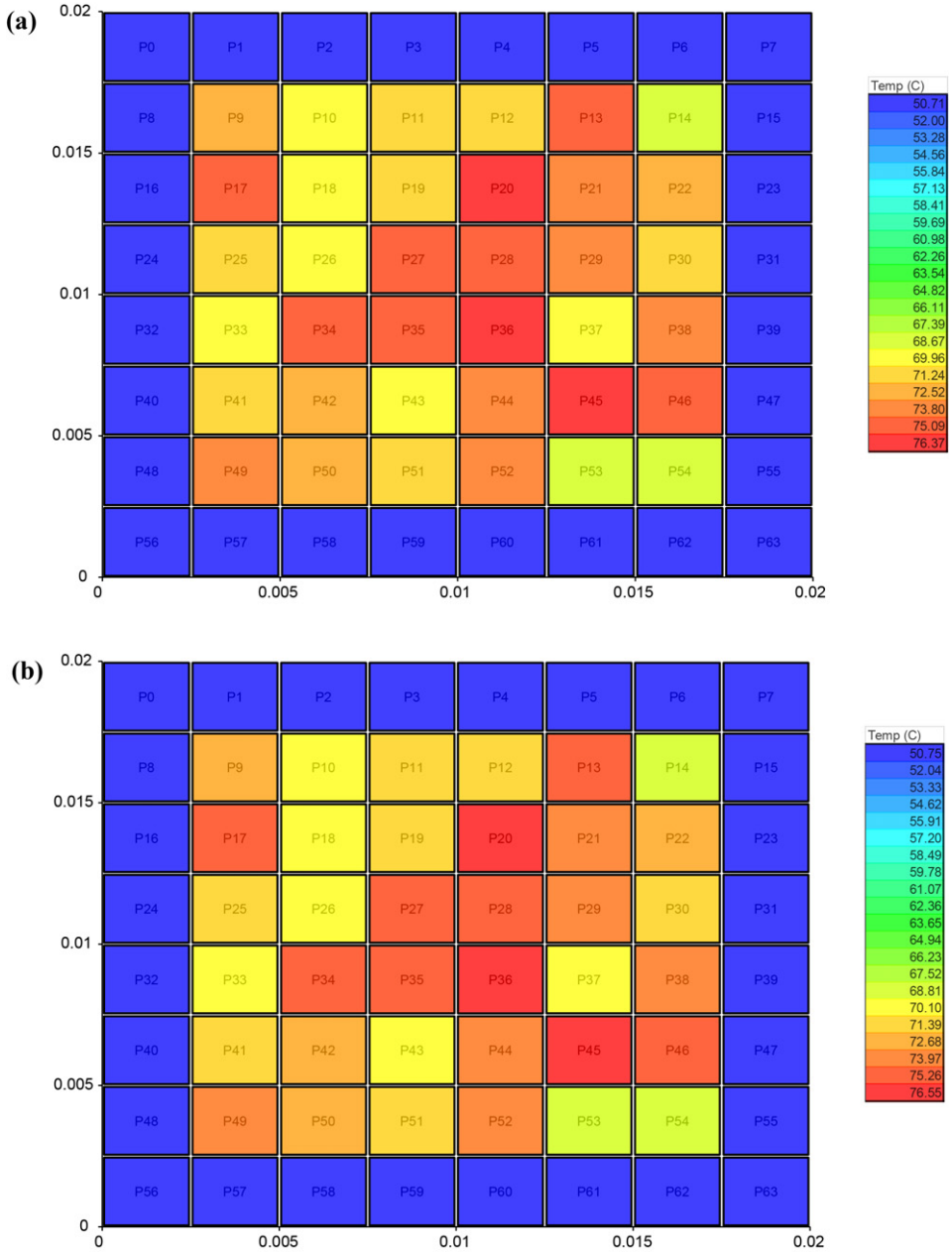

Fig. 10. Thermal profile of $N = 64$ systems with (a) *2D_parallel* and (b) *2D_parallel + wireless* architecture.

are denoted by *f6*, *f3* and *f2* respectively based on the computation resources (number of computation nodes) they need for execution. Based on the composition of jobs executing on our system, we bin the system job loads into two categories – one in which *f6* jobs are dominant and the other in which *f3* and *f2* jobs occupy up to half of all the nodes. The total number of jobs concurrently executing on the system is clearly higher in the latter case. Since each *f6* individually requires the largest number of computation nodes (six), the probability that one will be allocated a contiguous partition on the network is relatively low. Therefore, the above test plan represents the conservative end of the spectrum for performance evaluation.

4.2. Computation throughput

To measure the computation throughput of our system, we only use each basic operation (logarithm/exponentiation) performed by a core (see Section 3.1) as the unit (leaving out addition because it is much simpler), and the number of operations per second as the metric. Computation throughput is not only affected by the mix of jobs running on the system at any point of time, but also by allocation time overhead, usage of wireless shortcuts, and network architecture. Fig. 6 shows the computation throughput for the two different job loads mentioned earlier across different network architectures and system sizes. *2D_parallel + wireless* consistently provides the best computation throughput across job loads and system sizes. It is interesting to note that the best performing architecture has a wireless-agnostic allocation method. While wireless-aware allocation methods guarantee that a larger proportion of flits use the wireless shortcuts (see Fig. 7), this also leads to congestion over these links. Since we use a static routing technique that is based only on the comparison of distances traversed in alternative paths (using shortcuts vs. not using shortcuts), we end up routing more flits through the wireless shortcuts than their bandwidth can sustain without incurring a latency penalty. In

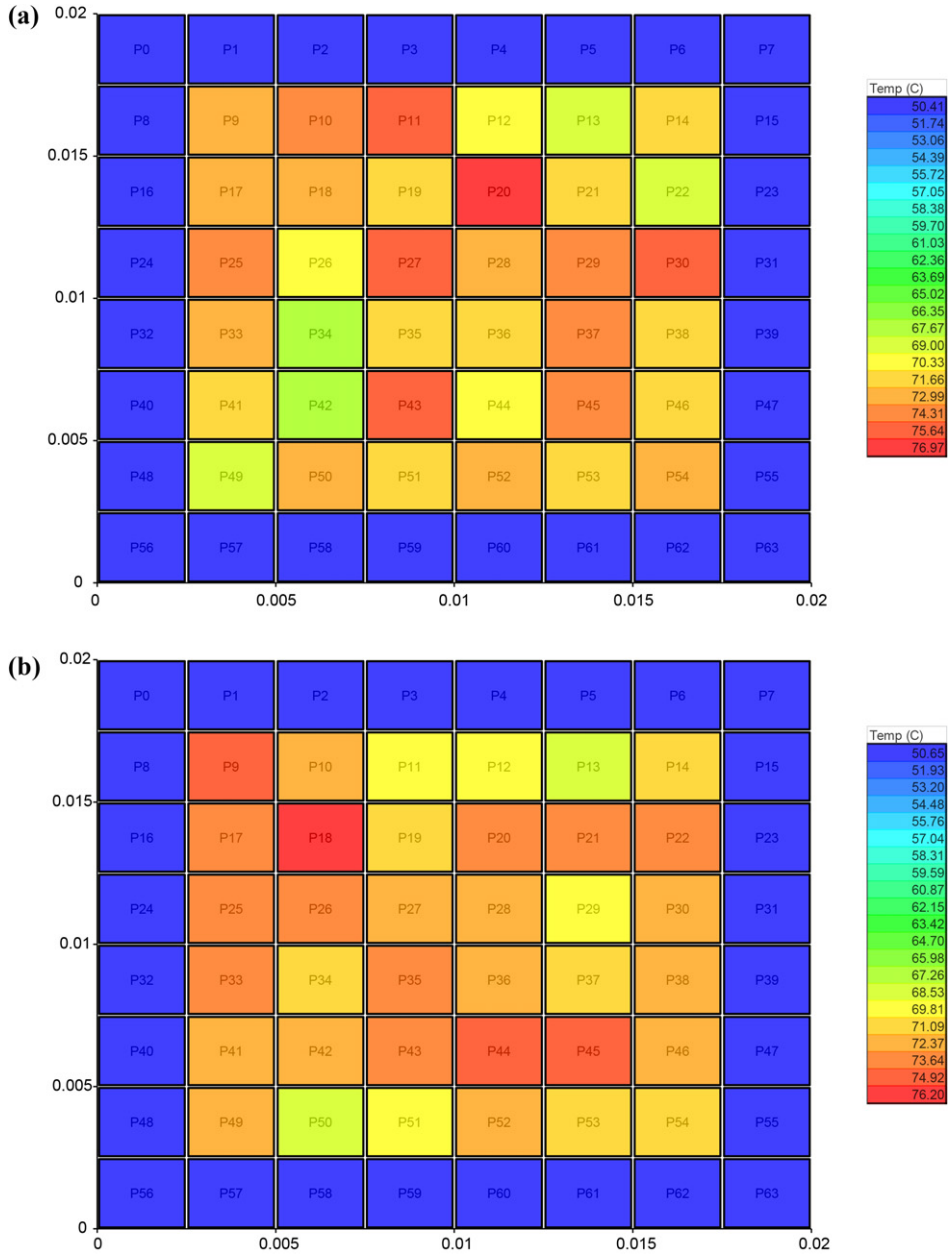

Fig. 11. Thermal profile of $N = 64$ systems with (a) *diameter wireless + Hilbert* and (b) *diameter wireless + column-major* architecture.

a wireless-agnostic allocation method such as *2D_parallel + wireless*, we try to maximize the number of A-type partitions during allocation, leaving to the wireless shortcuts the job of carrying traffic from B-type partitions.

Referring to Fig. 6, we also note that the cases containing a higher proportion of *f2* and *f3* jobs have a 5–10% higher computation throughput than the *f6* dominant loading scenario. Note that a larger system size (Fig. 6(b)) provides proportional gain in computation throughput because the problem size can be appropriately scaled up. The lowest parallelization efficiency is obtained for *wireless + column-major* and this is attributed to the high allocation-time overheads for larger system sizes, proving that this allocation method is less scalable with system size.

4.3. Proportion of flits using wireless shortcuts

Fig. 7 shows the percentage of total flits that used the wireless shortcuts. Note that the number of shortcuts (three) is much lower than the number of nodes (64, 256) in the system. As expected, *2D_parallel + wireless*, being a wireless-agnostic allocation method, leads to the lowest proportion of flits using wireless shortcuts. On the other hand, *wireless + column-major* allocation leads to the highest proportion of flits using wireless shortcuts across system sizes. This is because it is a wireless-aware allocation method, in which the partitions that do not get direct access to wireless shortcuts are aligned with the shortcuts, providing them with access to the shortcuts during routing, as explained earlier in Section 3.4.3.

4.4. Energy and power consumption

Average power dissipation in the chip is important from the physical perspective, because it is a direct indicator of the activity of the logic inside the chip and has a bearing on its thermal profile as explained further in Section 4.6. Quite predictably, the average power dissipation is higher in architectures that can deliver higher computation throughput, as shown in Fig. 8. In fact, wireless-aware

allocation consistently leads to lower average power dissipation. We have included a data point to show that wireless link placement under uniform random traffic assumption leads to even lower power dissipation for $N = 64$, although this is primarily because fewer computations are being performed and fewer messages are being transferred per second. Note that, the reduced power dissipation comes at the cost of reduced throughput performance in all cases. Consequently, we evaluated the energy consumption profiles of the architectures under consideration.

In order to determine which architecture is indeed the most energy-efficient, we evaluated the energy spent per operation. This consists of the computation energy component spent within the computation nodes, and the network energy component spent in the network switches, wireless transceivers and wired links. Fig. 7 shows a comparison of the energy spent per operation across different network architectures and system sizes. *2D_parallel + wireless* is the most energy-efficient in terms of overall energy consumption per operation. A closer look reveals that for $N = 64$, the network energy component is indeed lower for the wireless-aware methods, *wireless + Hilbert* and *wireless + column-major*, due to a larger proportion of their flits using wireless shortcuts, each of which consumes less energy than a wired link. However, due to higher computation latencies and the greater contribution of the computation energy component, the overall energy per operation turns out to be higher. For $N = 256$, the proportion of flits using wireless shortcuts is low across all architectures, and the saving in energy due to flits using wireless shortcuts is more than offset by the additional energy consumption in the wired links. This leads us to the conclusion that *2D_parallel + wireless* is still the best performing architecture from the energy-efficiency perspective.

4.5. Traffic statistics

In order to thoroughly analyze the throughput and energy-efficiency of different architectures, we need to understand the nature of traffic that our application generates. Our use-case model does not generate a deterministic traffic pattern. Hence, we try to characterize the traffic in terms of its first, second and third order statistical properties, and correlate these with throughput, energy consumption and power dissipation. A good indicator of traffic is the number of flits routed per network switch while running the application. We measure the mean, standard deviation and skew of this quantity across all 64 (256) switches for $N = 64$ ($N = 256$), as shown in Fig. 9. For $N = 64$, the mean values are about the same across architectures; for $N = 256$, *diameter wireless + column-major* clearly needs to route more flits per network switch, which indicates congestion and hence reduced throughput as we have seen earlier. Note that the standard deviation varies across architectures for both system sizes, and is the least for *2D_parallel + wireless*, which has the highest throughput and lowest energy per operation. Traffic is clearly less skewed for wireless-agnostic architectures than for wireless-aware architectures. This is attributable to congestion around shortcuts in wireless-aware architectures, as discussed earlier. Higher skew is strongly correlated with lower throughput and higher energy per operation. Following the discussion in Section 4.4, higher skew is also correlated with lower power dissipation owing to reduced network and PE activity.

4.6. Thermal profile

Thermal profiling of a many-core chip is important in order to prevent chip failure due to extreme temperatures during periods of peak activity. It is also important to ensure that a large number of hotspots are not created and on-chip temperature variation is low enough not to introduce timing failures. With this objective, we used HotSpot 5.0 [32] to carry out thermal profiling of our systems with $N = 64$ to determine the relationship between NoC architecture and on-chip thermal variation. As shown in Fig. 8, the majority of the power dissipation is due to computation activity in the PEs. Hence, the method of allocating these PEs to different jobs has a direct bearing on thermal variation and hotspot creation.

Fig. 10(a) and (b) respectively shows thermal profiles for *2D_parallel* and *2D_parallel + wireless* that look similar albeit for a slightly higher peak temperature in the latter case. Although the maximum temperature (<77 °C) is well within reliability limits, a clustering of hotspots is noticed. Compare this with the thermal profiles of systems having a wireless-aware architecture (Fig. 11). Since average power dissipation is lower in these cases, the thermal profile indicates a more even distribution of hotspots, although the on-chip range of temperature is similar to that seen in Fig. 10. This observation can be expected as we found the wireless-aware architectures to compromise on throughput and thus dissipate lower average power, which naturally translates to a lower probability of hotspot generation. The system designer has to decide the trade-off between higher, energy-efficient throughput on one hand, and lower propensity for hotspot creation on the other, while choosing the NoC architecture and PE allocation approach.

5. Conclusion

This paper proposes, designs and evaluates novel NoC-based hardware accelerator platforms targeted toward high-throughput scientific applications. The on-chip network is built using both wired links and on-chip wireless links, the latter being used as long-range shortcuts to further reduce inter-core message latency. In addition to achieving high-throughput, we show that our many-core accelerator platforms are energy-efficient.

We achieved computation throughput of over 10^{11} log/exp operations per second for a class of scientific applications involving concurrently-executing jobs of similar nature but variable computational footprint, while consuming ~0.5 nJ for each such operation. Our systems dissipate 55 W (for $N = 64$) and 213 W (for $N = 256$) and the maximum on-chip temperature is capped at 77 °C, demonstrating that high throughput is achieved without sacrificing energy-efficiency or exceeding power and thermal budgets, thereby being thermally efficient. We analyze the traffic behavior through statistical properties and correlate these with observations of throughput performance and power dissipation. We explore several NoC architectures and evaluate them with respect to the above-mentioned parameters and present design tradeoffs between throughput and energy-efficiency, and on-chip thermal variation. The results presented herein provide solutions to several challenges a system architect faces when designing low-energy high-performance many-core hardware accelerators.

References

[1] Amazon Elastic Compute Cloud. <http://aws.amazon.com/ec2/> (accessed 06.09.12).
[2] W.J. Dally, B. Towles, Route packets, not wires: on-chip interconnection networks, in: Proceedings of the Design Automation Conference, 2001, pp. 684–689.
[3] S.V. Tota, et al., A case study for NoC-based homogeneous MPSoC architectures, IEEE Transactions on VLSI Systems 17 (March (3)) (2009) 384–388.
[4] U.Y. Ogras, R. Marculescu, It's a small world after all: NoC performance optimization via long-range link insertion, IEEE Transactions on VLSI Systems 14 (July (7)) (2006) 693–706.
[5] A. Ganguly, et al., Scalable hybrid wireless network-on-chip architectures for multi-core systems, IEEE Transactions on Computers 60 (October (10)) (2011) 1485–1502.
[6] The CIPRES Science Gateway. <http://www.phylo.org/sub_sections/portal/> (accessed 06.09.12).
[7] National Center for Biotenchonology Information. <http://www.ncbi.nlm.nih.gov/guide/all/#tools_> (accessed 06.09.12).

[8] Earth System Modeling Framework. <http://www.earthsystemmodeling.org/> (accessed 06.09.12).
[9] A. Kalyanaraman, Algorithms for genome assembly, in: D. Padua (Ed.), Encyclopedia of Parallel Computing, Springer Science+Business Media LLC, New York, USA, 2011, http://dx.doi.org/10.1007/978-0-387-09766-4.
[10] T. Majumder, P. Pande, A. Kalyanaraman, Accelerating maximum likelihood based phylogenetic kernels using network-on-chip, in: Proceedings of 23rd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 26–29 October 2011, 2011, pp. 17–24.
[11] S. Sarkar, T. Majumder, A. Kalyanaraman, P.P. Pande, Hardware accelerators for biocomputing: a survey, in: Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS 2010, 30 May–2 June 2010, 2010, pp. 3789–3792.
[12] T. Oliver, et al., Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW, Bioinformatics 21 (16) (2005) 3431–3432.
[13] V. Sachdeva, et al., Exploring the viability of the cell broadband engine for bioinformatics applications, in: Proceedings of IEEE International Symposium on Parallel and Distributed Processing, 2007, pp. 1–8.
[14] M. Herbordt, et al., Single pass, BLAST-like, approximate string matching on FPGAs, in: Proceedings of 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2006, pp. 217–226.
[15] W. Liu, et al., Streaming algorithms for biological sequence alignment on GPUs, IEEE Transactions on Parallel and Distributed Systems 18 (9) (2007) 1270–1281.
[16] S. Sarkar, G.R. Kulkarni, P.P. Pande, A. Kalyanaraman, Network-on-chip hardware accelerators for biological sequence alignment, IEEE Transactions on Computers 59 (January (1)) (2010) 29–41.
[17] T.S.T. Mak, K.P. Lam, High speed GAML-based phylogenetic tree reconstruction using HW/SW codesign, in: Proceedings of Computational Systems Bioinformatics, 2003, p. 470.
[18] N. Alachiotis, et al., Exploring FPGAs for accelerating the phylogenetic likelihood function, in: Proceedings of IEEE International Symposium on Parallel and Distributed Processing, 2009, pp. 1–8.
[19] J.D. Bakos, P.E. Elenis, A special-purpose architecture for solving the breakpoint median problem, IEEE Transactions on VLSI Systems 16 (December (12)) (2008) 1666–1676.
[20] F. Blagojevic, et al., RAxML-Cell: parallel phylogenetic tree inference on the cell broadband engine, in: Proceedings of IEEE International Symposium on Parallel and Distributed Processing, 2007, pp. 1–10.
[21] F. Pratas, et al., Fine-grain parallelism using multi-core, cell/BE, and GPU systems: Accelerating the phylogenetic likelihood function, in: Proceedings of International Conference on Parallel Processing, 2009, pp. 9–17.
[22] T. Majumder, S. Sarkar, P. Pande, A. Kalyanaraman, NoC-based hardware accelerator for breakpoint phylogeny, IEEE Transactions on Computers 61 (June (6)) (2012) 857–869.
[23] T. Majumder, M. Borgens, P. Pande, A. Kalyanaraman, On-chip network-enabled multi-core platforms targeting maximum likelihood phylogeny reconstruction, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 31 (July (7)) (2012) 1061–1073.
[24] C. Kah-Hyong, R. Paramesran, B.H.S. Asli, L. Chern-Loon, Efficient hardware accelerators for the computation of Tchebichef moments, IEEE Transactions on Circuits and Systems for Video Technology 22 (March (3)) (2012) 414–425.
[25] A. Morad, T. Morad, L. Yavits, R. Ginosar, U. Weiser, Generalized MultiAmdahl: optimization of heterogeneous multi-accelerator SoC, Computer Architecture Letters (2013), http://dx.doi.org/10.1109/L-CA.2012.34.
[26] B.-G. Nam, H. Kim, H.-J. Yoo, Power and area-efficient unified computation of vector and elementary functions for handheld 3-D graphics systems, IEEE Transactions on Computers 57 (April (4)) (2008) 490–504.
[27] Circuits Multi-Projects, 46, Avenue Félix Viallet, 38031 Grenoble, France. <http://cmp.imag.fr> (accessed 06.09.12).
[28] R. Marculescu, U.Y. Ogras, L.-S. Peh, N.E. Jerger, 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 (January (1)) (2009) 3–21.
[29] K. Kempa, et al., Carbon nanotubes as optical antennae, Advanced Materials 19 (2007) 421–426.
[30] D. Hilbert, Über die stetige Abbildung einer Linie auf ein Flächenstück, Mathematische Annalen 38 (3) (1891) 459–460.
[31] P.P. Pande, et al., Performance evaluation and design trade-offs for network-on-chip interconnect architectures, IEEE Transactions on Computers 54 (August (8)) (2005) 1025–1040.
[32] W. Huang, K. Sankaranarayanan, R.J. Ribando, M.R. Stan, K. Skadron, Accurate, pre-RTL temperature-aware processor design using a parameterized, geometric thermal model, IEEE Transactions on Computers 57 (September (9)) (2008) 1277–1288.
[33] A. Stamatakis, RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models, Bioinformatics 22 (November (21)) (2006) 2688–2690.
[34] The Exelixis Lab, Heidelberg Institute for Theoretical Studies, Heidelberg, Germany. <http://sco.h-its.org/exelixis/software.html> (accessed 06.09.12).

Turbo Majumder is a doctoral candidate at the School of Electrical Engineering and Computer Science at Washington State University, Pullman, USA. He received his B.Tech. (Hons.) in Electronics and Electrical Communication Engineering and M.Tech. in Automation and Computer Vision, both from Indian Institute of Technology, Kharagpur, India in 2005. His research interests include network-on-chip and multi-core system-on-chip design for biocomputing applications, VLSI design, and parallel and high-performance computer architectures. Prior to joining the PhD program, he worked for Freescale Semiconductor and Nvidia Graphics. He is a Student Member of the IEEE.

Partha Pratim Pande received the M.S degree in Computer Science from the National University of Singapore and the Ph.D. degree in Electrical and Computer Engineering from the University of British Columbia, Vancouver, BC, Canada. He is an Associate Professor at the School of Electrical Engineering and Computer Science, Washington State University, Pullman. His current research interests are novel interconnect architectures for multicore chips, on-chip wireless communication networks, and hardware accelerators for biocomputing. He has around 70 publications on these topics. Dr. Pande currently serves on the Editorial Board of IEEE Design and Test of Computers and Sustainable Computing: Informatics and Systems. He also serves in the program committee of many reputed international conferences. He has won the NSF CAREER award for his research on WiNoCs in 2009. He is a Senior Member of IEEE.

Ananth Kalyanaraman is an Associate Professor at the School of Electrical Engineering and Computer Science in Washington State University and an affiliate faculty at Molecular Plant Sciences. He received his BE degree from Visvesvaraya National Institute of Technology, India (1998), and his MS degree (2002) and Ph.D degree (2006) from Iowa State University. His research interests lie at the intersection of computational biology and parallel processing. He is a recipient of a DOE Early Career Research Award, two best paper awards and serves on a number of conference organizing and program committees. He is a member of ACM, IEEE/CS, ISCB and SIAM.