High-Performance and Energy-Efficient Network-on-Chip Architectures for Graph Analytics

KARTHI DURAISAMY, HAO LU, PARTHA PRATIM PANDE,
and ANANTH KALYANARAMAN, Washington State University at Pullman, Pullman, WA

With its applicability spanning numerous data-driven fields, the implementation of graph analytics on multicore platforms is gaining momentum. One of the most important components of a multicore chip is its communication backbone. Due to inherent irregularities in data movements manifested by graph-based applications, it is essential to design efficient on-chip interconnection architectures for multicore chips performing graph analytics. In this article, we present a detailed analysis of the traffic patterns generated by graph-based applications when mapped to multicore chips. Based on this analysis, we explore the design-space for the Network-on-Chip (NoC) architecture to enable an efficient implementation of graph analytics. We principally consider three types of NoC architectures, viz., traditional mesh, small-world, and high-radix networks. We demonstrate that the small-world-network-enabled wireless NoC (WiNoC) is the most suitable platform for executing the considered graph applications. The WiNoC achieves an average of 38% and 18% full-system Energy Delay Product savings compared to wireline-mesh and high-radix NoCs, respectively.

CCS Concepts: • **Mathematics of computing → Graph algorithms**; • **Hardware → Radio frequency and wireless circuits**; **Network on chip**

Additional Key Words and Phrases: Graph analytics, wireless NoCs, community detection, graph coloring

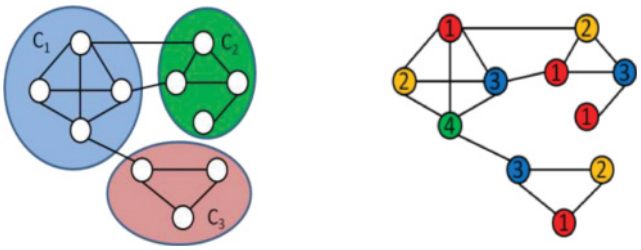
ACM Reference Format:
Karthi Duraisamy, Hao Lu, Partha Pratim Pande, and Ananth Kalyanaraman. 2016. High-performance and energy-efficient network-on-chip architectures for graph analytics. ACM Trans. Embed. Comput. Syst. 15, 4, Article 66 (August 2016), 26 pages.
DOI: http://dx.doi.org/10.1145/2961027

1. INTRODUCTION

The prevalence of multicore architectures opens new opportunities of running data-parallel applications on a single chip instead of using large clusters. In an era when power constraints and data movement are proving to be significant barriers for the application of high-end computing, multicore architecture offers a low-power and high bandwidth platform suitable for data-intensive applications. The Network-on-Chip (NoC) paradigm has emerged as a revolutionary methodology for integrating a very

This article is an extended version of the following conference paper that appeared in CASES 2015.
Duraisamy, K., Lu, H., Pande, P. P., and Kalayanaraman, A. 2015. High performance and energy efficient wireless NoC-enabled multicore architectures for graph analytics. In *Proceedings of 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*. pp. 147–156.
This work was supported in part by the US National Science Foundation (NSF) grants CCF-0845504, CCF-1514269, and CCF-1162202; an Army Research Office grant W911NF-12-1-0373; as well as US DOE award DE-SC-0006516.
Authors' addresses: K. Duraisamy, H. Lu, P. Pande, and A. Kalyanaraman, The School of Electrical Engineering and Computer Science, Washington State University, EME 102, 355, Spokane Street, Pullman, WA-99164-2752; emails: {kduraisa, hlu, pande, ananth}@eecs.wsu.edu.

DOI: http://dx.doi.org/10.1145/2961027



(a) Community Detection (b) Graph Coloring

Fig. 1. Illustrative examples for the two different graph operations addressed in this article – viz. community detection and graph coloring. Given a graph $G(V,E)$, (a) shows a possible community-wise partitioning of vertices in V, and (b) shows a 1-distance coloring of vertices, with the numbers within each vertex identifying the color assignment.

high number of embedded cores in a single die. Hence, performance of NoC-enabled multicore chips needs to be evaluated for emerging big data applications.

There is some effort in designing multicore chips customized for emerging big-data workloads [Schadt et al. 2010]. Many data-driven applications use complex graph representations. Achieving parallel scalability in graph applications remains a significant challenge due to the inherent irregularity in real-world networks that in turn causes irregularities in computation and data movement. Consequently, mapping graph-theoretic applications on modern-day multicore architectures designed with low-latency and energy-efficient NoC will be key to executing large-scale graph operations efficiently. Towards this goal, in this work, we explore state-of-the-art Wireless NoC (WiNoC)-enabled multicore architectures for efficient implementation of advanced graph operations. As exemplars of advanced graph analytics, we focus on two graph operations – community detection and graph coloring. Figure 1 shows an illustrative example for the two operations. We have chosen these not only because they encapsulate many of the key graph-algorithmic traits (irregular computation, data movement, locality issues, etc.), but also because they embody two different paradigms that are prevalent in many of the advanced parallel graph processing frameworks. The first paradigm is that of applications that perform vertex-centric calculations followed by synchronized updates. The second paradigm is to apply synchronization during the vertex-centric calculation phase. These find applicability in a number of other graph operations such as page-rank, graph matching, and clustering.

Given an undirected input graph $G(V,E,\omega)$, where V is the set of vertices, E is the set of edges, and ω is a function that maps every edge to a numeric weight, the problems underlying the two target graph operations can be stated as follows:

Community Detection. The community detection problem [Fortunato 2010] is one of partitioning the set of vertices in V into "communities" such that the *modularity* of the partitioning [Newman 2006] is maximized. *Modularity* is a measure, between 0 and 1, that reflects the quality of partitioning. More specifically, it is the ratio of between the net weights of the intra-community edges to inter-community edges. Neither the number of communities nor the size distribution of communities is known *a priori*. In fact, community detection is used to reveal such natural divisions that exist in real-world networks. It is used in a number of scientific applications including (but not limited to) social network analysis, bioinformatics, collaboration networks, and electric power grid [Fortunato 2010]. Despite its broad application base, executing community detection over large real-world graphs remains to be a challenging problem despite recent developments in multicore processing. The heuristic nature of algorithms alongside the need to access neighborhoods of vertices in an irregular fashion causes irregular data

access and movement patterns that impede performance and scalability in traditional multicore environments.

Balanced Coloring. The classical problem of graph coloring can be stated as follows: Given an input graph $G(V,E)$[1], assign colors to vertices such that no two vertices that share an edge between them are assigned the same color [Jensen and Toft 1995]. Coloring is a classical graph operation that is widely used in a number of graph-based scientific applications to determine compatible parallel schedules [Leighton 1979; Jones and Plassmann 1993]. As edges in graphs typically represent vertex-to-vertex interdependencies, coloring can be used to obtain a parallel schedule that guarantees no two vertices that are interdependent on one another are processed during the same parallel step (i.e., same "color"). However, such an approach needs to also have as few parallel steps (or color classes) as possible, and therefore a second goal for graph coloring is to minimize the number of colors used in assignment. In addition, since concurrency is limited by the number of vertices within each color class, there is also a need to ensure load-balanced color distribution and this variant has been extensively studied under the context of equitable and balanced coloring [Furmanczyk 2004; Bodlanendrea and Fominb 2005]. Recently, coloring implementations have been proposed to obtain an initial coloring so as to minimize the number of colors and then redistribute the vertices among color classes so as to obtain a balanced coloring [Lu et al. 2015a]. The assignment of colors to vertices and their redistributions make the graph-coloring operation data movement-bound and lock-intensive in traditional multicore environments. In addition, these characteristics heavily depend on the underlying graph structure and connectivity.

The irregular data access patterns and memory-bound nature of community-detection operation, and the communication-bound and lock-intensive nature of the graph-coloring operation represent two unique challenging use-cases for multicore parallelism. In this article, we undertake a design-space exploration of various NoC architectures to enable efficient implementations of above-mentioned graph operations. Specifically, we consider three NoC topologies, traditional mesh, small-world network [Ogras and Marculescu 2006; Wettin et al. 2014] and a high-radix network-like Flattened Butter-Fly [Kim et al. 2007; Sewell et al. 2012]. We analyze the computation and on-chip traffic patterns generated by different phases of community detection and balanced graph-coloring applications on real-world graphs and determine the most suitable NoC architecture for these applications. Our analysis shows that, among all the NoC architectures considered here, the small-world NoC architecture enabled by long-range wireless shortcuts achieves the best overall Energy Delay Product (EDP).

2. RELATED WORK

2.1. Graph Analytics Algorithms and Architectures

Designing specialized computation architectures and parallel algorithms for big data analytics has been an area of great interest in recent times. The Blue Gene architecture [Chen et al. 2012] is one such architecture for distributed memory systems, enabling more than 10^{15} floating-point operations per second. Blue Gene also enables efficient interconnection of thousands of computing nodes through its advanced 5D torus interconnection topology. A detailed study on using micro-architectures for graph analytics has been presented in Ediger [2013]. In Bader et al. [2005], the performance of Symmetric Multiprocessor (SMP) systems for large-scale graph analytics has been analyzed. There are several previous works exploring the efficiency of SMPs

[1]Edge weights are of no consequence to typical graph-coloring problems. Only the connectivity between vertices matter.

for applications with inherent irregularities [Castro et al. 2013; Francesquini et al. 2015; Frasca et al. 2012]. For the two target graph operations of community detection and graph coloring, multithreaded implementations for traditional multicore (x86) architectures and also many core architectures such as Tilera have been previously developed [Lu et al. 2015b; Chavarría-Miranda et al. 2014; Çatalyürek et al. 2012; Reidy et al. 2012; Staudt and Meyerhenke 2013]. In Wu et al. [2011], a MapReduce-based parallel computing model for large graph analytics has been presented. The model has been shown to be particularly efficient for two popular graph operations, 3-clique enumeration of a graph and computation of clustering coefficient.

2.2. Network-on-Chip

As stated in Section 1, NoCs have emerged as standard communication backbones for multicore chips. Conventional NoCs employ a multi-hop, packet-switched communication where each computing core is connected to a NoC router. Today's industry standard multicore platforms such as Tilera Gx-72 [Tilera Corporation 2015], Adapteva Epiphany [Gwennup et al. 2011], mainly follow a wireline mesh interconnection topology. In such systems, the network latency is usually high due to the inherent multi-hop nature of the network. It is already shown that, by inserting long-range shortcuts in a regular mesh architecture to induce small-world effects, it is possible to achieve significant performance gain and lowered energy dissipation compared to traditional multi-hop mesh networks [Ogras and Marculescu 2006; Marculescu et al. 2009]. The concept of express virtual channels is introduced in Kumar et al. [2008]. By using virtual express lanes to connect distant cores in the network, it is possible to avoid the router overhead at intermediate nodes, and thereby improve NoC performance in terms of power, latency and throughput. Performance is further improved by incorporating ultra low-latency multi-drop on-chip global lines (G-lines) for flow control signals [Krishna et al. 2008]. Kalray's MPPA architecture [De Dinechin et al. 2014] uses a modified 2D torus NoC topology (MPPA-NoC) and is specifically designed to establish low hop data transfers between processing cores and the I/O terminals located along the chip edges. Like any folded torus topology, the MPPA-NoC uses physically long metal wires spanning between the opposite edges of a chip in order to reduce the average inter-router hop counts.

Despite significant performance gains, the long-range links in the above schemes are designed with conventional wires. It is already shown that, beyond a certain length, wireless links are more energy efficient than the conventional metal wires [Deb et al. 2012]. Hence, the performance improvements achieved by using long-range wireless links will be more than that using wired links. The viability of on-chip wireless communication has been already demonstrated through prototype developments [Lin et al. 2007; Zhang et al. 2007; Seok and O 2005; Branch et al. 2005]. However, in order to harvest its full potential, more research is required to address various challenges in several areas including system architecture, circuit design, device fabrication, and CAD tool development [Deb et al. 2012]. A comprehensive survey regarding various wireless NoC (WiNoC) architectures is presented in Deb et al. [2012], shows the possibility of creating novel architectures by inserting on-chip wireless links. Robust, nature-inspired, small-world network-based wireless NoC architectures are presented in Deb et al. [2013] and Wettin et al. [2014]. These architectures employ mm-wave wireless links operating in the 10GHz-100GHz range as long-range shortcuts and are shown to outperform the traditional mesh NoC architecture in terms of network latency and energy dissipation. It should be noted that photonic links could also be used as long-range shortcuts in the small-world NoCs. However, the implementation of optical NoCs still faces major challenges integrating with standard CMOS electronics [Bogaerts et al. 2014]. One of the major advantages using mm-wave wireless link is that it is CMOS compatible and no new technology is required to implement this.

Algorithm 1: Community Detection $G(V,E,\omega)$

repeat ▸For each phase
  Init: $C(v) \leftarrow \{v\}, \ \forall v \in V$
  repeat ▸For each iteration
    for each $v \in V$ **do** in parallel
      //Vertex-centric Computation
      $C_l(v) \leftarrow$target community that maximizes modularity gain for v
      // Synchronized Updates
      $C(v) \leftarrow C_l(v), \forall v \in V$
      C.info ← *Update community sizes*
  until (Net Modularity Gain < τ_1)
  G'(V', E', ω') ← *compact G(V,E,*ω*))* using C
until (Net Modularity Gain < τ_2)
Output final communitites

(a) Community detection algorithm

Algorithm 2: Balanced Coloring $G(V,E)$,*Colors*

$C_f \leftarrow \{ c \mid c \in Colors$ and *sizeof* $(c) > \tau \}$
for each $c \in C_f$ **do**
  for each $v \in c$ **do** in parallel
    // Synchronized Updates and Computation
    Color(v) ← select an underfull color
    Colors.info ← Update source & target colors
Output final (balanced) colors

(b) Balanced coloring algorithm

Fig. 2. Pseudocodes for graph algorithms.

The design of low-hop-count high-radix on chip networks are presented in Kim et al. [2007], Abeyratne et al. [2013], and Sewell et al. [2012]. In these high-radix networks, multiple computing cores are connected to a single NoC router and hence a large number of ports are required in each router. Connecting multiple cores to a single router leads to a lower number of NoC routers and hence enables a low-average hop-count among the communicating nodes. The Flattened Butter-Fly (FBF) topology [Kim et al. 2007; Sewell et al. 2012] is one such high-radix topology where the maximum hop count between the computing cores can be limited to two. Though high-radix topologies provide a low-hop-count network, designing efficient routers, incorporating large number of router ports is a challenging task [Krishna et al. 2013]. In Sewell et al. [2012], it is demonstrated that, by employing *sizzle-switches*, it is possible to design four stage routers for the Flattened Butter-Fly topology operating at 3GHz in 32nm technology node. Recently, reconfigurable asynchronous NoC architectures incorporating wires with clockless repeaters have been proposed in Krishna et al. [2013, 2014]. These NoC designs, referred as SMART NoCs, restrict the operating frequency of NoC to 1GHz - 2GHz, involve high control overheads and require more complex router designs than the traditional NoC routers.

In this work, we undertake an exhaustive design-space exploration for NoC architectures on shared memory platforms, to support optimized implementation of advanced graph analytics. To represent a wide-range of NoC architectures, we consider traditional mesh, emerging high-radix and small-world network-enabled WiNoC in this performance evaluation and establish the relevant design tradeoffs.

3. INTRODUCTION TO COMMUNITY DETECTION AND COLORING

3.1. *Community Detection: Grappolo*

Recently, Lu et al. [2015b] developed a scalable parallel implementation for executing community detection on conventional multicore architectures. This method, called *Grappolo*, implements a multiple-phase, multiple-iteration heuristic to maximize the modularity of the output partitioning. Figure 2(a) presents the pseudocode for *Grappolo*. In what follows, we outline the algorithmic details and data structures of *Grappolo*.

Given an input graph $G(V,E,\omega)$, containing n vertices and m edges, the algorithm executes multiple phases and multiple iterations within each phase, until convergence. Within each phase, the following steps are carried out:

(1) Initially, each vertex is assigned to an individual community of its own.
(2) Within each iteration, the vertices are scanned in parallel, and for each vertex, a decision is made to determine whether or not to migrate it to one of its neighboring communities (as defined by the communities of its neighbors). To avoid locking, the parallel implementation uses community assignments as of the previous iteration. Other graph heuristics are used to resolve potential conflicts and in effect to ensure that the net modularity gain achieved by the end of the iteration always remains non-negative.
(3) The algorithm proceeds to subsequent iterations until the gain achieved in net modularity becomes negligible. Reaching convergence by this criterion marks the end of current phase and the algorithm constructs a compacted graph $G'(V',E',\omega')$ by collapsing every community detected in G to a single meta-vertex in G', and creating edges reflecting the weights of intra- and inter-community links in G.
(4) Subsequently, the algorithm initiates the next phase on the newly compacted graph G', until no more appreciable modularity gain is achieved.

The main data structures used in *Grappolo* are as follows: The graph is stored as two arrays of size $O(m+n)$ in the compressed sparse row (CSR) format [Saad 2003]. Each vertex entry also stores the current community assignment for that vertex. In addition, a separate array data structure is used to keep track of the degree of each community (i.e., sum of the degree of all vertices belonging to each community). The orders in which the vertices and communities are stored in their respective arrays are arbitrary as it is not possible to predetermine locality properties due to the dynamic nature of the algorithm.

3.2. Balanced Coloring

Given an input graph $G(V,E)$, the goal of 1-distance coloring is to determine a color assignment to vertices such that adjacent vertices are always assigned different colors. A classic greedy coloring heuristic used for minimizing the number of colors used can be described as follows [Çatalyürek et al. 2012]: For each vertex $v \in V$ (in some order), assign a color that is not yet used by any of its neighboring vertices. Typically, the first available color index is used for this purpose (assuming the colors are identified numerically), and if all colors are used by the neighbors, a new color is introduced. This heuristic represents a Greedy First-Fit approach and has been shown in practice to typically create a highly skewed color size distribution.

In a recent work, Lu et al. [2015a] proposed a number of heuristics to generate a balanced coloring – i.e., the number of vertices assigned per color class (aka color bin) is roughly the same. They implemented a parallel approach whereby the Greedy First Fit approach is used to first obtain an initial coloring of the graph, and a subsequent balancing step *redistributes* the vertices among the different bins so as to obtain a balanced coloring without increasing the overall number of colors used. Different balancing schemes were explored and one of the more effective schemes, namely *Greedy-CLU*, works as follows (see Figure 2(b) for a pseudocode). Color classes are processed one at a time, such that the vertices within every over-full color bin (i.e., those containing more than the mean number of vertices) are redistributed to one or more under-full bins until the number of vertices within the source bin reaches the target mean or no more valid vertex moves are possible. The choice of the under-full bin is performed greedily by selecting a smallest available bin that is also compatible with the vertex being moved (i.e., contains no neighbors of that vertex).

This implementation also uses the CSR format for storing the input graph, while a separate array-based data structure is used to keep track of the color bins and their sizes. The parallel scalability of the overall coloring process is affected by the need to lock the coloring data structures in order to update the source and destination bin sizes. In addition, the need to perform compatibility checks for vertices requires access to color bins corresponding to the neighbors of every vertex. This step introduces irregular memory access and workloads.

Thus, the balanced coloring application involves two distinct phases, an *initial coloring phase* and a *redistribution phase*, between which the redistribution phase is a communication-bound phase with heavy vertex migration traffic and locking. Furthermore, the distribution phase is devoid of any complex computations. Hence, unlike the initial coloring phase, the execution speed of the redistribution phase is highly dependent on the speed of the data migration.

Henceforth, for ease of exposition, we will refer to the above two specific multicore algorithms for community detection and balanced coloring as simply "*Grappolo*" and "Balanced Coloring", respectively. These two algorithms will serve as our target implementations for the WiNoC-based multicore platform.

4. NOC ARCHITECTURES FOR GRAPH ANALYTICS

Both community detection using *Grappolo* and Balanced Coloring applications exhibit inherent irregularities due to data movements. As stated in Section 3, the computation within *Grappolo* has multiple phases with alternating parallelized (clustering) and serialized (compaction) characteristics. In addition, *Grappolo* involves migration of vertices between the communities during each clustering phase. In contrast, balanced coloring consists of two major parallel phases, initial coloring and redistribution, with bulk of data movement and locking-related traffic occurring during redistribution. Due to these characteristics, the two applications generate substantial long-range traffic patterns when run on a multicore platform (as later shown in Figure 6(a)), with significant amount of data exchanges involve physically far apart cores. Furthermore, these applications show the presence of one or more hotspot nodes whose traffic injection rates are much higher than that of the average traffic injection rate (as later shown in Figure 6(b)). Due to these long-range and skewed traffic patterns, we posit that designing an on-chip interconnect infrastructure that enables low-latency data exchange, even among physically distant cores will be critical to achieve performance at scale in these graph applications.

Considering these facts, in this work we undertake a detailed performance evaluation of three NoC architectures, viz., WiNoC, mesh with long-range shortcuts, and Flattened Butter-Fly for *Grappolo* and Balanced Coloring. First, we present a brief tutorial regarding the salient features of these NoC architectures and then we will present the experimental results comparing and contrasting performance of these architectures for graph analytics. It should be also noted that we focus our investigation mostly on the overall interconnection architecture aspect of the NoCs. Hence, we omit the SMART NoC (discussed in Section 2) from this performance evaluation. The performance advantages of SMART mainly comes from the asynchronous interconnect and the SMART control mechanism. This principle can be applied to any NoC topology and that is out of scope of this work.

4.1. WiNoC

In the proposed WiNoC topology, each core is connected to a router; routers are interconnected using wireline and wireless links. The topology of the WiNoC is a small-world network where the links between routers are established following a power law distribution. More precisely, the probability $P(i,j)$ of establishing a link between two routers

i and j, separated by a Euclidean distance l_{ij}, is proportional to the distance l_{ij} raised to a finite power as in:

$$P(i, j) = \frac{l_{ij}^{-\alpha} f_{ij}}{\sum_{\forall i} \sum_{\forall j} l_{ij}^{-\alpha} f_{ij}}. \quad (1)$$

The frequency of traffic interactions between cores, f_{ij}, is also factored in, so the more frequently communicating cores have a higher probability of having a direct link inserted between them. This frequency is the percentage of traffic generated by core i that is sent to core j directly. This approach implicitly optimizes the network architecture for a non-uniform traffic scenario.

Getting into details, the parameter α governs the nature of connectivity, e.g., a larger α means a locally connected network with a few or even no long-range links. By the same token, a zero value of α generates an ideal small-world network following the Watts-Strogatz model [Watts and Strogatz 1998] – one with long-range shortcuts that are virtually independent of the distance between the cores. It has been shown that a value of α less than $D + 1$, D being the dimension of the network, ensures the small-world property; with $\alpha = 1.8$, the average hop count is minimum with a fixed wiring cost [Petermann and De Los Rios 2005]. Figure 3(a) shows an example Small World Network on Chip (SWNoC) architecture designed following (1) and is constituted only of metal wire links. As it is evident from this figure, the SWNoC uses several long metal wires.

Since the long metal wires are costly both in terms of power and latency, we propose to use wireless links to connect the routers that are far apart in our small world based WiNoC architecture (shown in Figure 3(b)). In practice, depending upon the available wireless resources, we can only allow a limited number of long links in the WiNoC to be wireless, while the others would still remain wireline. This way, we can make the distant cores "socialize" with each other, and hence reduce the communication costs when running real applications. We use the mm-wave wireless links operating in the 10GHz – 100GHz range here. In Deb et al. [2013], it is demonstrated that it is possible to create three non-overlapping wireless channels with on-chip mm-wave wireless transceivers. Using these three channels, we overlay the wireline small-world connectivity with the wireless links such that a few routers get an additional wireless port. The wireless port of each router is provided with Wireless Interface (WI) tuned to one of the three distinct frequency channels. The WI placement mechanism that is used to overlay the power-law-based wireline connectivity with wireless nodes is explained later in Section 4.4.

4.1.1 Routing Protocol. The power-law model-based WiNoC principally has an irregular network topology. Irregular networks require topology agnostic routing methods. Hence, in this work, we use ALASH protocol [Wettin et al. 2014] for routing packets in WiNoC. ALASH is built upon the layered shortest path (LASH) algorithm. The LASH algorithm takes advantage of the multiple virtual channels in each port of the NoC routers in order to route messages along the shortest physical paths. In order to achieve a deadlock-free operation, the network is divided into a set of virtual layers, which are created by dedicating the virtual channels from each router port into these layers. The shortest physical path between each source-destination pair is then assigned to a layer such that the layer's channel-dependency graph remains free from cycles and this ensures the deadlock freedom. ALASH protocol improves on the LASH routing scheme, by enabling an adaptive layering function by considering the expected traffic patterns. We follow the *priority* layering function explained in Wettin et al. [2014]. The priority layering function allocates as many layers as possible to source-destination pairs with high f_{ij}. This improves the adaptability of messages with higher f_{ij} by providing them with greater routing flexibility. In ALASH, if the WI

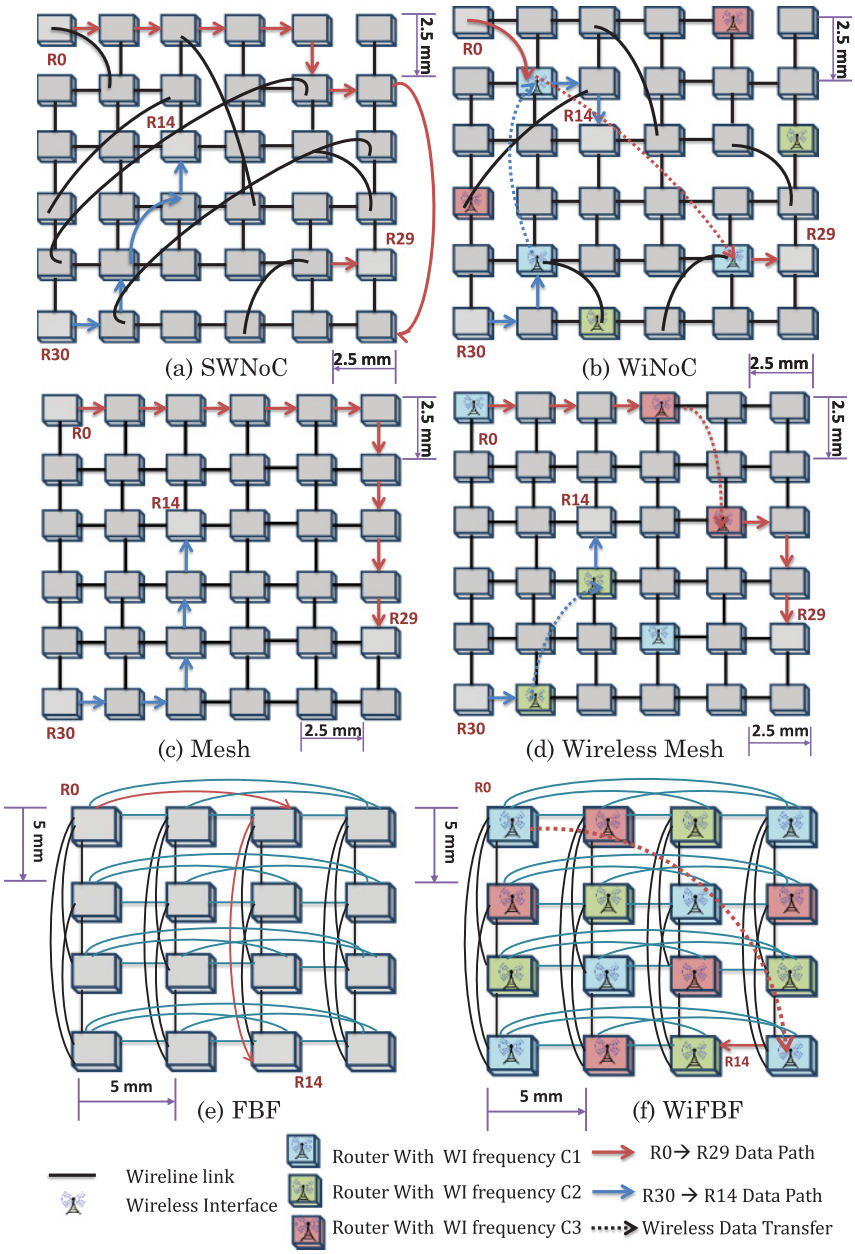


Fig. 3. Various NoC architectures showcasing data transfers. Figures 3(a)–3(d) showcase 36 NoC routers with 2 data transfers (R0→R29 &R30→R14). Figures 3(e) and 3(f) showcase FBF topologies with 16 routers and one data transfer (R0→R14). In wireless enabled NoCs, the color of the router indicates the wireless channel frequency that is associated with the router.

located in the path becomes unavailable due to lack of buffer space or due to failures, then the packet is rerouted from the WI along wireline only paths [Wettin et al. 2014].

4.2. Mesh

As stated earlier, mesh NoC architecture is today's industry standard due to various advantages like simple design, regular structure, easy timing closure, and the like. However, as stated in Section 2, standard Mesh NoCs (example in Figure 3(c)), are not capable of handling long-range traffic efficiently, as they involve multi-hop transmissions. Hence, we consider the mesh architecture with long-range shortcuts here.

4.2.1. Wireless Mesh. In Ogras and Marculescu [2006], a mechanism to enhance the performance of the wireline mesh architecture by employing long-range shortcuts is demonstrated. Following [Ogras and Marculescu 2006], in this work, we augment the wireline mesh architecture with long-range wireless shortcuts and this NoC architecture is called as Wireless Mesh (WiMesh) NoC. The architecture is shown in Figure 3(d). Here also we use the mm-wave wireless links as long-range shortcuts.

4.2.2. Routing in Wireless Mesh. The mesh NoC incorporating long-range wireline shortcuts presented in Ogras and Marculescu [2006] follows a distributed routing methodology, referred to as *South-Last* Routing. In this mesh, NoC incorporating long-range wireline shortcuts, all the wireline ports (including those used for long-range connections) are assigned with particular directions such as north, east, south-east, north-west, and so on [Ogras and Marculescu 2006]. In *South-Last* Routing method, the routers without any long-range link follow the traditional XY routing methodology whereas the routers with long-range links follow a turn-restricted routing methodology. In this turn-restricted routing methodology, all the deadlock-inducing turns such as 180-degree turns are prohibited [Ogras and Marculescu 2006]. Comparing the directions assigned to the router ports identifies these restricted turns. However, in WiMesh, it is not possible to assign a particular direction to the Wireless Interface ports. Hence, employing the turn-restricted routing methodology for routers with WIs is not feasible. In this work, we follow a modified version of *South-Last* routing for the routers with WIs. In this modified version of *South-Last* routing, whenever wireless traversal is involved, we prevent deadlocks by allowing the message paths to follow a strictly increasing (or decreasing) router IDs. It is already shown in Dally and Seitz [1987], that a routing methodology where the message path follows a strictly increasing (or decreasing) router IDs is deadlock-free. In the case of wireless node failure, we will resort to standard X-Y wireline routing.

4.3. Flattened Butter-Fly (FBF)

The Flattened Butter-Fly (FBF) topology is a low-hop-count, high-radix topology. In this work, we consider a 16 router FBF topology presented in Kim et al. [2007], Sewell et al. [2012]. As shown in Figure 3(e), each router in FBF is connected to all other routers in its row and to all other routers in its column. Hence, in this FBF, all the NoC routers in the system can communicate with each other within two hops by following a dimension ordered XY routing [Sewell et al. 2012]. As it can be noted in Figure 3(e), the FBF topology incorporates a number of long-range wireline connections that can be costly in terms of delay and energy. In order to efficiently handle the long-range communication, in this work, we incorporate Wireless Interfaces in each of the NoC routers present in FBF. In WiFBF, the overall number of routers is a small fixed number (only 16) and hence even when each router is provided with a WI, the additional on-chip area overhead is relatively low. This modified network is called as Wireless enabled Flattened Butter-Fly (WiFBF) and the connectivity is shown in Figure 3(f). Similar to the WiMesh and WiNoC, we use the mm-wave wireless channels here.

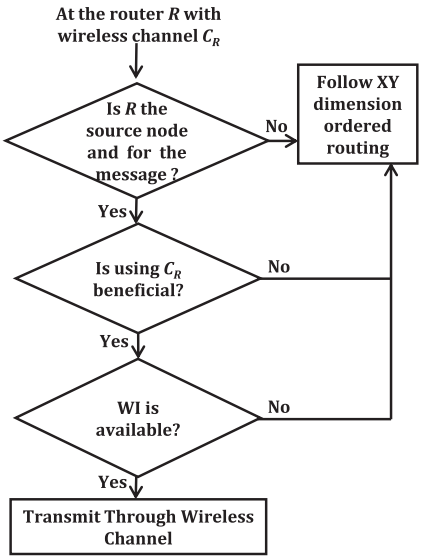


Fig. 4. WiFBF routing.

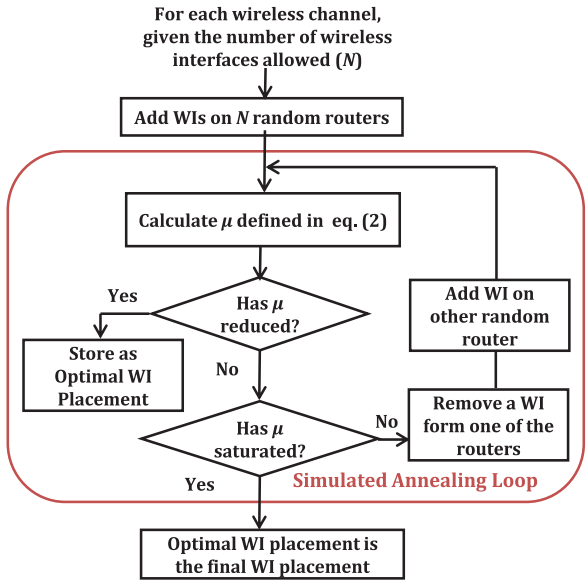


Fig. 5. Wireless interfaces placement.

4.3.1. Routing Protocol. In Sewell et al. [2012], to route the packets along the wireline FBF network, a minimal XY dimension ordered routing scheme (with a maximum inter router hop count of two) is used. In this work, we expand on this scheme to design the routing methodology for WiFBF. The WiFBF routing protocol followed in a router R with a WI channel frequency of C_R, is shown in Figure 4. As shown in Figure 4, in WiFBF routing, each message is allowed a maximum of one wireless traversal and the wireless routing for any message can be only initiated at the source router for that message. Hence, there is no opportunity for creating cycles and thus the WiFBF routing is deadlock-free. Under WI failures (or when a WI is unavailable), the proposed WiFBF routing simply follows the minimal XY-dimension-ordered routing along the wireline links and hence the maximum hop count for any communication in WiFBF is only two.

4.4. Placement of Wireless Interfaces

As discussed earlier (Section 4.3), in the WiFBF architecture, each router is provided with a Wireless Interface. However, in WiMesh and WiNoC, based on the allowed area overhead, we can only incorporate a limited number of WIs and hence it is essential to optimize the location of these WIs to ensure performance gain without paying high area overhead. The flowchart presented in Figure 5 shows a simulated annealing-based mechanism used for placing a set of N WIs on each wireless channel in WiMesh and WiNoC. In the WiMesh and WiNoC creation process, the wireless interface placement strategy focuses on minimizing the optimization metric μ, which is defined as:

$$\mu = \Sigma_{\forall i} \Sigma_{\forall j} f_{ij} h_{ij}. \tag{2}$$

Here h_{ij} denotes the minimum distance in number of hops from router i to router j with the given network connection. The f_{ij} value denotes the frequency of interaction between the routers. The f_{ij} values are determined by analyzing the traffic patterns generated while executing the *Grappolo* and Balanced Coloring applications with a set of graphs. To optimize the network for handling multiple applications, a combined f_{ij} metric can be used. Let L denote the number of applications considered. Then, the net

interaction frequency is given by:

$$f_{ij} = \frac{1}{L} \sum_{n} \frac{f_{ijk}}{\sum_{\forall i} \sum_{\forall j} f_{ijk}}, \quad k = 1, 2 \ldots L. \tag{3}$$

where f_{ijk} denotes the frequency of interaction between routers i and j for a specific application k. As elaborated later in Section 5.1, the f_{ij} of an application has a unique signature with constant traffic hotspots that can be observed over all different input graphs. Hence, any set of large real-world graphs can be used to identify the f_{ij} patterns. Moreover, as shown later in Figure 7, these hotspots communicate almost evenly with all the other nodes and do not involve in any high-intensity pairwise communication. Hence, placing Wireless Interfaces (WIs) to minimize the overall traffic weighted hop count would ensure the following:

(1) The WIs would be placed nearer to the traffic injection hot spots
(2) The remaining WIs would be essentially distributed all across the chip such that the overall inter-router hop count is minimized.

It has been already shown that wireless links are more energy efficient than traditional metal wires only when the link length exceeds 7.5mm in the 65nm technology node [Deb et al. 2013]. Hence, in addition to the simulated annealing mechanism explained in Figure 5, we also impose a WI placement constraint where two WIs operating in the same frequency channel are separated by more than 7.5mm.

Following the mechanism explained in Figure 5 and by varying N, we can find the optimum number of wireless interfaces for a NoC and their best suitable locations. The experimental results showcasing the optimum value of N for both WiMesh and WiNoC are discussed later in Section 5.2.

4.5. Wireless Medium Access Control (MAC) Protocol

For all the three wireless-enabled NoCs (WiNoC, WiMesh, and WiFBF) discussed above, we employ a distributed MAC protocol to resolve the channel access contentions among the wireless nodes [Duraisamy et al. 2015]. The wireless channel used in our wireless NoCs is a shared medium. Hence, each wireless node inherently knows if there is any on-going wireless transmission. If a wireless node has messages to transmit and if the wireless channel is free, the wireless node first broadcasts a request to acquire the channel. The request packets follow a simple orthogonal on-off keying mechanism. This enables multiple wireless nodes to simultaneously transmit the request packets. Once the requests are received, all the nodes process the requests simultaneously and one of the WIs acquires the wireless channel in the next cycle [Duraisamy et al. 2015].

4.6. Wireless Interface

The two principal wireless interface components for the WiNoC architecture are the antenna and the transceiver. WiNoC uses a metal zigzag antenna that has been demonstrated in Deb et al. [2013], as it provides the best power gain with the smallest area overhead. A detailed description of the transceiver circuit is out of the scope of this article. However, the transceiver was designed and fabricated and all the details are provided in Wettin et al. [2014]. The wireless interface is completely CMOS compatible and no new technology is needed for its implementation. The wireless interface has an area overhead of 0.25mm^2 per transceiver in 65nm technology node. Considering a 20mm$\times$20mm die, the addition of wireless interfaces gives rise to 0.93%, 1.125%, and 1% silicon area overheads for WiMesh, WiNoC, and WiFBF architectures, respectively, for a 64-core system. It should be noted that the area overhead varies depending on the specific NoC architecture due to variation in the number of wireless interfaces (as

explained later in Section 5.2). However, the overall additional area overhead is still negligible. We use the three non-overlapping on-chip wireless channels demonstrated in Deb et al. [2013], with operating frequencies of 31GHz, 57.5GHz, and 120GHz. Each WI is tuned to one of these three frequencies. For data rates of 16Gbps, these wireless link dissipates 1.95pJ/bit for a 20mm communication range.

5. EXPERIMENTAL RESULTS AND EVALUATION

Test platforms. In this section, we evaluate the performance of a 64-core multiprocessor system running the *Grappolo* and Balanced Coloring applications. We consider the WiMesh, WiNoC, and WiFBF as the primary NoC architectures for this performance evaluation. Additionally, we also consider the wireline counterparts of all the wireless architectures, viz., the traditional wireline mesh, a fully wireline small-world NoC (SWNoC) and FBF. We use GEM5 [Binkert et al. 2011], a full-system simulator, to obtain detailed processor and network-level information. We consider a system of 64×86 processors running Linux within the GEM5 platform for all the experiments. The application codes for Grappolo and Balanced Coloring are parallelized for multicore platforms using OpenMP multithreading with dynamic scheduling. The tasks are parallelized in term of vertices without involving any optimization based on data locality and hence the considered applications are highly scalable with increasing system sizes [Lu et al. 2015a, 2015b]. In GEM5 simulation, the parallelized tasks are mapped to the emulated parallel threads using basic Linux scheduler. The memory system is MOESI_CMP_directory setup with private 64KB L1 instruction and data caches and a shared 16MB (256KB distributed per core) L2 cache. The width of all wireline links is considered to be the same as the flit width, which is considered to be 32 bits in this article.

The die size of the system under consideration is $20 \times 20 mm^2$ and we consider a system frequency of 2.5GHz (for both cores and the routers), with 65nm technology nodes. For mesh and small-world architectures, we employ a generic three-stage router architecture explained in Pande et al. [2005]. For high -adix architectures, we employ four-stage router architectures as suggested in Sewell et al. [2012]. For the small-world NoCs, we consider an average of four ports per router, which is similar to that of the conventional wireline mesh. Also in small-world NoCs, we impose an upper bound on the number of ports attached to a particular router so that no router becomes unrealistically large. The SWNoC and WiNoC achieve highest throughput with lowest energy dissipation when the maximum port count in a router is restricted to seven [Wettin et al. 2014]. As stated earlier, the FBF and WiFBF architectures are having 16 routers and hence each NoC router is connected to 4 computing cores. For all the NoC architectures considered, all the router ports are provided with a buffer depth of two flits. Energy dissipation of the NoC routers, inclusive of the routing and MAC blocks, was obtained from the synthesized netlist by running Synopsys^{TM} Prime Power, while the energy dissipated by wireline links was obtained through HSPICE simulations taking into consideration the length and layout of wired links. In all the NoC architectures, across both wireline and wireless links, we follow wormhole routing methodology. The processor-level statistics generated by the GEM5 simulations are incorporated into McPAT (Multicore Power, Area, and Timing) to determine the processor-level power values [Li et al. 2009].

Test inputs. The evaluation of *Grappolo* is performed with the help of DIMACS10 clustering instance graph data sets [DIMACS10 2016]. We have considered five DIMACS10 clustering instance graphs, Hep-th (HEP), Astro-ph (ASTRO), Comd-mat-2003 (COND), PGPgiantcompo (PGP), and as-22july06 (ASJ), to evaluate the benefits achieved by the WiNoC interconnect in execution of *Grappolo* applications. The

Table I. DIMACS10 Graphs Used for *Grappolo* Analyses

Graph	Graph Size	Number of Vertices	Number of Edges	Number of Communities	WiNoC Execution Time (Seconds)
ASJ	0.46 MB	22,963	48,436	31	1.588099
PGP	0.25 MB	10,680	24,316	99	0.284018
ASTRO	1.2 MB	16,706	121,251	412	1.724108
HEP	0.154 MB	8,361	15,751	636	0.292018
COND	1.3 MB	31,163	120,029	945	2.035129

Table II. DIMACS10 Graphs Used for Balanced Coloring Analyses

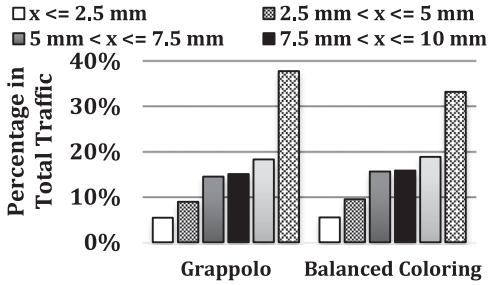
Graph	Graph Size	Number of Vertices	Number of Edges	Number of Output Colors	WiNoC Execution Time (Seconds)
NTH	34.9 MB	2,216,688	2,441,238	5	1.628099
BLG	21.3 MB	1,441,295	1,549,970	5	1.244078
CCR	14.6 MB	268,495	1,156,647	17	0.332021
CNR	35.1 MB	325,557	2,738,969	86	0.372023
CAC	9.8 MB	227,320	814,134	87	0.436028
CAD	12.1 MB	299,067	977,676	115	0.290009

evaluation of the Balanced Coloring application is carried out with the help of six DIMACS10 graphs. Two street network graphs, Belgium (BLG) and Netherlands (NTH), three citation network graphs, citationCiteseer (CCR), coAuthorsDBLP (CAD), coAuthorsCiteseer (CAC), and one clustering instance graph cnr-2000 (CNR) are used for these evaluations. Due to the shorter runtimes for the Balanced Coloring application, we were able to test much larger data sets. Further information regarding the datasets such as, size, vertex and edge counts, number of communities and colors and application execution times are provided in Tables I and II.

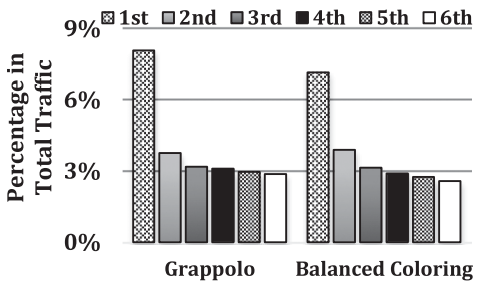
5.1. Analysis of Traffic Patterns

In this section, we analyze the nature of the traffic patterns generated by *Grappolo* and Balanced Coloring when mapped on to the 64-core system considered here. Figure 6(a) shows the percentage of total traffic exchanged between the communicating cores separated by certain distances for *Grappolo* and Balanced Coloring. From this figure, we can observe that both *Grappolo* and Balanced Coloring generate a significant amount of long-range traffic, although owing to different reasons. In case of *Grappolo*, long-range traffic is largely due to shared memory access during the clustering computation; while in the case of Balanced Coloring, it is due to locks generated during the vertex redistribution phase. For both considered applications, it is evident that about 70% of the total injected messages are transferred among cores that are more than 7.5mm apart. As stated earlier, it is more energy efficient to use wireless links than metal wires for link length exceeding 7.5mm in the 65nm technology node. Hence, for this long-range communication, wireless links will be more beneficial.

Figure 6(b) shows the percentage of the traffic injected by each of the top six traffic hotspots present in the system, while executing *Grappolo* and Balanced Coloring. It can be observed from this plot that up to 10% of the total traffic is associated with a single core. Moreover, the top 6 traffic hotspots in *Grappolo* are responsible for about 25% of the total injected traffic. In Balanced Coloring, the top 6 traffic hotspots are responsible for up to 27% of the total system traffic. In case of *Grappolo*, this injection originates mostly from the master core, which is responsible for the graph compaction phase. Recall that the compaction step is largely serialized owing to the need to gather vertices belonging to communities. In case of Balanced Coloring, the hotspots appear during the redistribution phase. This can be attributed to the overheads associated

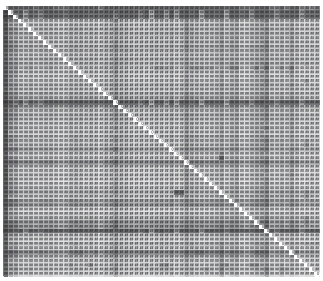


(a) Distance between communicating cores

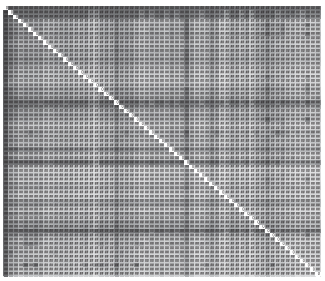


(b) Traffic injections from top 6 hot spots

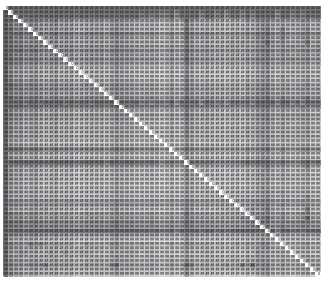
Fig. 6. Traffic characteristics. (a) Fraction of total traffic exchanged between the communicating cores separated by distance x for *Grappolo* and Balanced Coloring. The values for x are specified in the legend. (b) Percentage in total traffic injected by the top 6 traffic hotspots for *Grappolo* and Balanced Coloring.



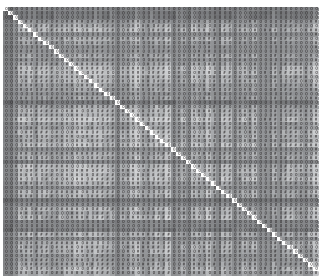
(a) *Grappolo*- PGP



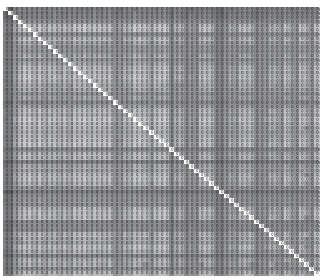
(b) *Grappolo*-ASTRO



(c) *Grappolo*- COND



(d) Balanced Coloring- CAD



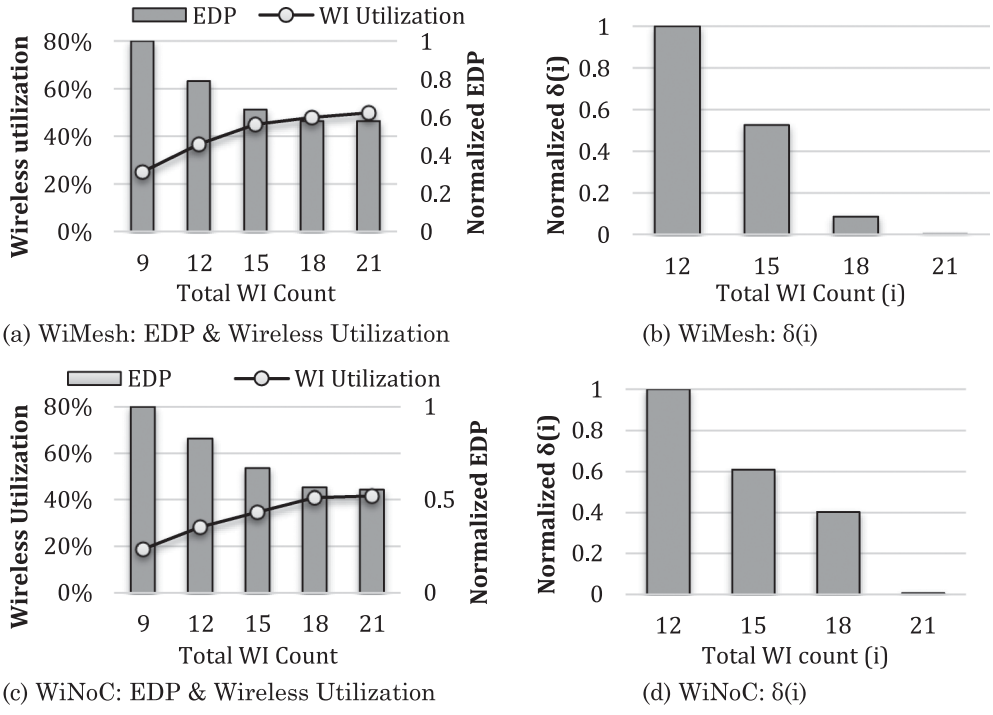
(e) Balanced Coloring- NTH

(f) Top left corner of Fig. 7(e)

Fig. 7. Heat Plots Representing f_{ij} matrices (of order 64×64) for Grappolo and Balanced Coloring with Different Input Graphs. Figure 7(f) shows the zoomed version of the top left corner in Figure 7(e) along with the source and destination node IDs. These heat plots show the relative intensities of all the elements in a f_{ij} matrix. The elements in a row represent the traffic intensity form a particular source to all the destination nodes in the system and thus the hotspots are indicated with darker bars in Figures 7(a) to 7(d). It can be noted that for an application, the hotspots remain the same across all the graphs.

with the management of locks. During the redistribution phase, the vertices processed are themselves located randomly on the network, effectively distributing the traffic injection rates at the sources. However, when multiple vertices located at different cores compete to acquire locks, the lock management introduces hotspots during acquisition and release of those locks.

The above-mentioned long-range and hotspot traffic patterns are observed consistently over all the input graphs considered here. To elaborate, Figures 7(a)–(c) and 7(d)–(e) show the traffic patterns for Grappolo and Balanced Coloring considering different data sets respectively. Figures 7(a)–(e), indicate the relative intensities of all



(a) WiMesh: EDP & Wireless Utilization

(b) WiMesh: δ(i)

(c) WiNoC: EDP & Wireless Utilization

(d) WiNoC: δ(i)

Fig. 8. Determining the Optimum Number of Wireless Interfaces.

the elements in a 64×64 f_{ij} matrix. The arrangement of source and destination nodes in these f_{ij} matrices can be understood from Figure 7(f). As shown, the elements in each row of this f_{ij} matrix represent the intensity of the traffic from a particular source node to all the destination nodes in the system. From Figure 7, it is evident that the probability of data-exchange between two given cores (i.e., f_{ij}) does not significantly vary across datasets but only across applications.

Thus, the above discussion indicates that both *Grappolo* and Balanced Coloring are characterized by significant long-range communication patterns and traffic hotspots. Thus, both present an ideal case for benefiting from the WiNoC architecture which is shown to efficiently handle the long-distance and hotspot traffics [Wettin et al. 2014].

5.2. Characterizing NoC Architectures

In this section, we characterize the network performance of all the NoC architectures considered. For these characterizations, we consider the combined graph analytic inter-core traffic pattern (f_{ij}), obtained with Equation (3) in Section 4.1.1. Since we are interested in both network latency and energy dissipation, we use the Energy-Delay Product (EDP) as the relevant performance metric. Average message latency and average message energy values are used in this message EDP computation.

5.2.1. Optimum Number of Wireless Interfaces. We first quantify the maximum number of WIs that can be added in WiNoC and WiMesh architectures, beyond which the addition of anymore WIs will provide no further significant improvement in performance. Figures 8(a) and 8(c) show the variation in EDPs of the WiMesh and WiNoC architectures, respectively, with increasing number of WIs. We also show the improvement achieved in wireless utilization (U) with increasing number of WIs. The wireless utilization parameter represents the percentage of total messages that are using the wireless

channels. To find the optimum number of WIs, we use a metric δ which is defined below.

$$\delta\,(\mathrm{i}) = \frac{U\,(i) - U(i - n_c)}{U(i - n_c)} \times \frac{EDP\,(i - n_c) - EDP(i)}{EDP(i - n_c)} \tag{4}$$

In the above equation, n_c represent the number of wireless channels available and i denote the number of WIs in the current system. We consider a step value of n_c in the above difference equations, since we add n_c wireless node in each step (one more WI per channel per step). The number of wireless interfaces (i) beyond which the δ approaches zero, can be considered as the optimum WI count.

Figures 8(b) and 8(d) show the δ(i) values for WiMesh and WiNoC architectures respectively. From Figures 8(a) and 8(c), it can be seen that the achievable EDP and wireless utilization improves until the WI count reaches a certain limit (18 in this case for WiNoC and 15 for WiMesh) and then remains constant. WiNoC has a better wireline connectivity to distribute the long-range traffic when compared to WiMesh NoC. Due to the inherent multihop characteristics of the baseline mesh network of WiMesh, more messages try to access the wireless paths to improve the latency. Hence, the wireless channels in the WiMesh system saturate with a lower number of WIs. Thus, we can conclude that the optimum number of WIs for WiNoC is 18 and the optimum number of WIs for the WiMesh is 15, beyond which the δ(i) values are near zero. Hence, from here on, for all the following experiments, we consider a set 15 WIs for WiMesh (5 WIs for each wireless channel) and 18 WIs for WiNoC (6 WIs operating for each wireless channel). As already discussed in Section 4 and as shown in Figure 3(f), WiFBF follows a fixed WI placement and channel assignment with WIs placed on all of 16 routers (Six WIs on channel 1, five WIs on channel 2 and five WIs on channel 3).

5.2.2. Network Performance. In this section, we present the comparative analysis of the network performances of all the NoC architectures considered in this work. As shown in Figure 9(a), the WiFBF and FBF architectures have the two lowest average inter-router hop count values. However, each router traversal in the FBF topology is costlier in terms of energy and latency, compared to that in mesh and small-world NoC topologies. This is further corroborated by the single router traversal energy values presented in Figure 9(b). Having a much larger number of ports in each router and requiring a hierarchal switching stage contribute to the additional router stage latency and energy experienced by FBF and WiFBF architectures.

The average message latency and EDP values are presented in Figures 9(c) and 9(d), respectively. From these figures, it is evident that all the wireless enabled NoC architectures perform better than the corresponding wireline only NoC following the same topology. Among all the topologies, mesh topology achieves the best latency and EDP improvement with the addition of wireless interfaces. This fact can be attributed to the high reduction in average hop count achieved by the mesh NoC with the addition of WIs (Figure 9(a)). Among all the wireline only NoCs considered, the FBF NoC achieves the lowest message latency. Similarly, among all the wireless-enabled NoCs, WiFBF achieves the best message latency value. However, due to their high router energy consumptions, in terms of EDP, the WiFBF and FBF topologies perform considerably worse than the WiNoC and SWNoC topologies (38% EDP penalty for WiFBF when compared to WiNoC), respectively. Aside from achieving the best network EDP values, the WiNoC exhibits only a 4.7% higher latency values when compared to WiFBF. From these analyses, we can conclude that for efficient implementations of complex large-scale graph analytics, the WiNoC architecture is the best suitable option.

5.2.3. Robustness of the NoC Architectures – Handling Runtime WI Failures. Any good wireless NoC architecture should be able to handle the runtime WI failures as seamlessly as

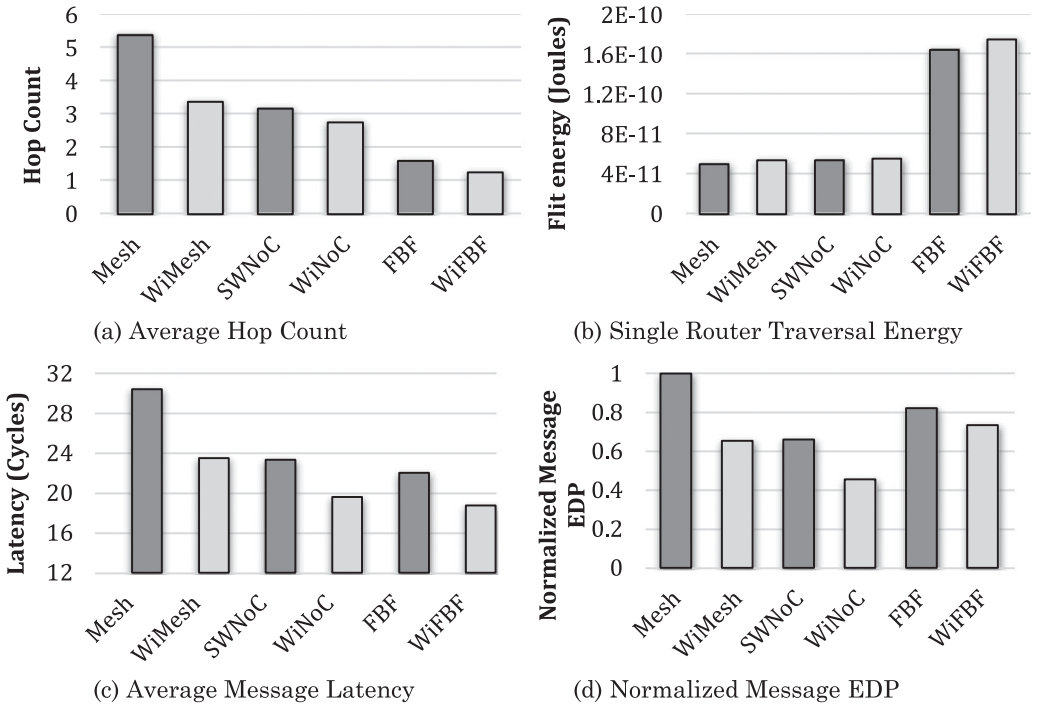


Fig. 9. Comparative analyses of network performances. Dark shaded bars indicate the Wireline only NoC performances while lighter shaded bars indicate the performances of the wireless enabled NoC architectures.

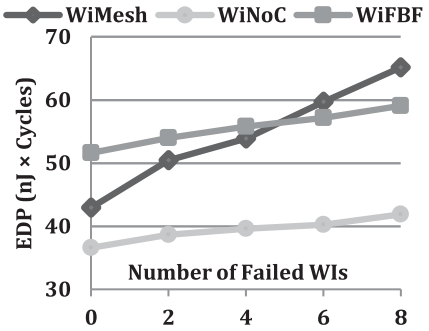


Fig. 10. NoC robustness. Increase in EDP against increase in number of failed WIs.

possible. As we have already explained in Section 4, in all three wireless NoCs (WiMesh, WiNoC and WiFBF), we handle the runtime WI failures by rerouting the packets along the wireline only paths. Figure 10 shows the increase in the EDP of the three above-mentioned wireless NoCs with increasing number of failed wireless nodes (WIs). From this figure, it can be easily seen that compared to WiNoC and WiFBF, the EDP of the WiMesh architecture increases more rapidly with the increase in number of failed WIs. This steeper increase can be attributed to the fact that the wireline only paths (used for rerouting) in the WiMesh topology have much higher average hop counts than that for the WiNoC and WiFBF topologies. In WiFBF, the maximum hop count is always two (even under WI failures) and the small-world topology of WiNoC ensures a low-average hop count even among the physically far apart routers. It can be seen from

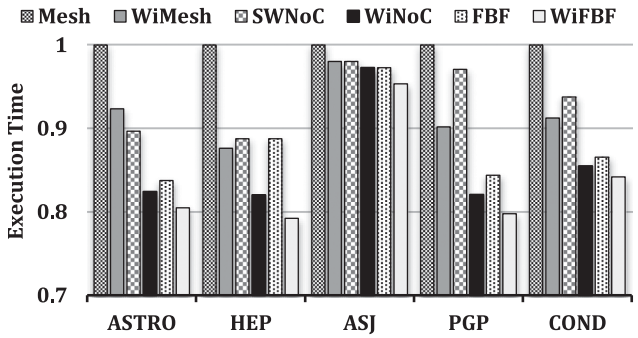


Fig. 11. *Grappolo*: Execution times. Shown values are normalized with respect to mesh NoC execution times.

Figure 9(a), that in fact small-world architecture with no wireless interfaces (SWNoC) has a smaller hop count than that of mesh architecture with 15 WIs (WiMesh). From these discussions, we can conclude that the WiNoC and WiFBF architectures are better in handling the runtime WI failures when compared to the WiMesh architecture.

5.3. Execution Time Improvements

In this section, we evaluate the overall runtime of all the NoC architectures considered when running the two target graph applications.

5.3.1 Grappolo. Figure 11 shows *Grappolo*'s execution times. Among all the NoC architectures considered, WiFBF and WiNoC achieve the lowest execution times due to their low hop count nature. When compared to the traditional wireline mesh, FBF achieves an average of 16.2% execution time improvement while WiNoC achieves an average of 14.2% execution time improvement. Among the five graphs considered, HEP achieves the highest execution-time improvement by using the WiNoC (17.9%) and WiFBF (20.5%), when compared to the wireline mesh. HEP is followed by ASTRO and PGP (with 19.5% improvement with WiFBF and 17.2% with WiNoC), when compared to the wireline mesh. The ASJ graph achieves the least execution time improvements (4.6% improvement with WiFBF and 3% improvement with WiNoC, when compared to the wireline mesh). The reason for the variations in execution-time improvement can be understood by analyzing the runtime characteristics of *Grappolo*. Figure 12 shows the varying traffic volumes and the average CPU utilizations, over the entire *Grappolo* execution period with the WiNoC architecture. Here, traffic volumes indicate the volume of flits exchanged through the interconnection network. As discussed earlier, the *Grappolo* application involves multiple executions of two distinct phases, clustering and compaction. Since all computing cores are highly active during the clustering phase, it is identified with high-average core utilizations. However, the master core is the only core that is active during compaction. This serialization within the compaction phase is identified with a lower average CPU utilization.

Since all the cores are active, inter-core traffic volumes associated with the clustering phase are much higher than that of the compaction phase. It can also be noted from Figure 12, that the traffic volume decreases with time during the clustering phase. As the execution of the community detection algorithm progresses, the number of clusters reduces, and consequently the interactions between the cores also begin to reduce, leading to the observed drop in traffic volume. Similar trends were observed with all the other NoC architectures albeit with different execution times (Figure 11).

These variations in CPU utilization and network traffic for the above-mentioned input graphs lead to the observed variation in runtime improvements achieved with

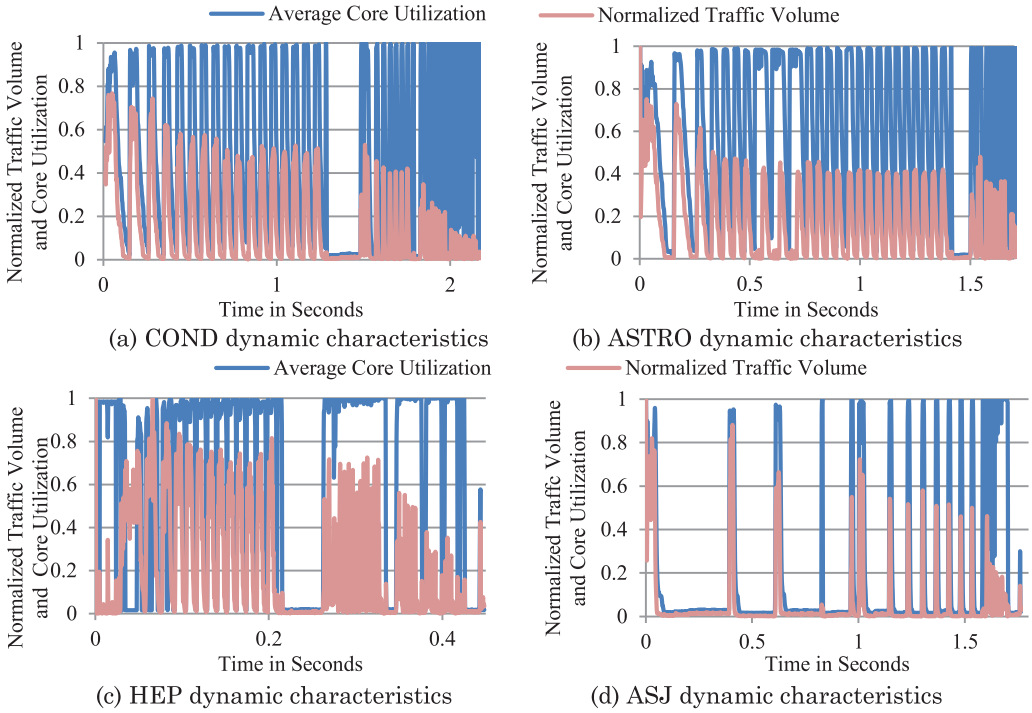


Fig. 12. *Grappolo* runtime characteristics: Variation in traffic volume and average CPU activity over multiple clustering and compaction phases with WiNoC interconnection architecture.

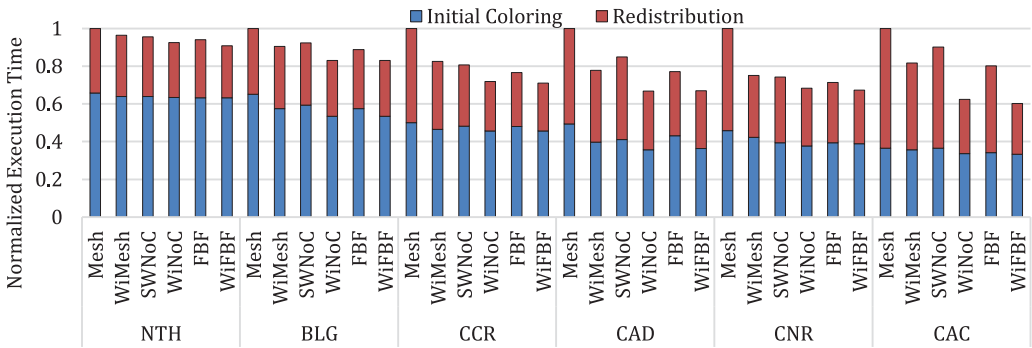


Fig. 13. Balanced coloring: Execution times.

different NoC architectures. By comparing Figures 11 and 12, we can note that the graphs with longer clustering durations achieve higher execution time improvements through the use of low hop count NoC architectures such as WiNoC, FBF and WiFBF. When compared to wireline mesh, the low hop count NoCs enable a faster data exchange among the computing cores, leading to a quicker executions of the traffic-intensive clustering phase.

5.3.2 Balanced Coloring. Figure 13 shows the execution times of the Balanced Coloring application on the multicore system interconnected with the six different NoC architectures. These execution times are broken down into the two major phases within

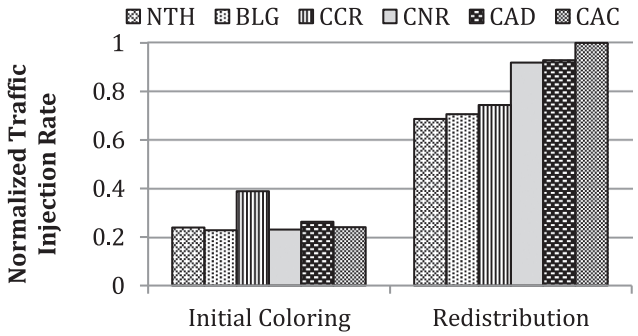


Fig. 14. Balanced coloring: Traffic injection rates.

computation – viz. initial coloring and redistribution. Among the graphs considered, CAC achieves the maximum improvement in execution time (38% by using the WiNoC and 40% with WiFBF, when compared to the wireline mesh). CAD follows CAC ((32%) by using the WiNoC and 33% with WiFBF when compared to the wireline mesh). The two street network graphs, BLG and NTH, achieve the least execution-time variations with varying NoC architectures. NTH achieves only a 7.5% execution-time improvement by using the WiNoC and only 9.3% with execution-time improvement by using WiFBF, when compared to using wireline Mesh.

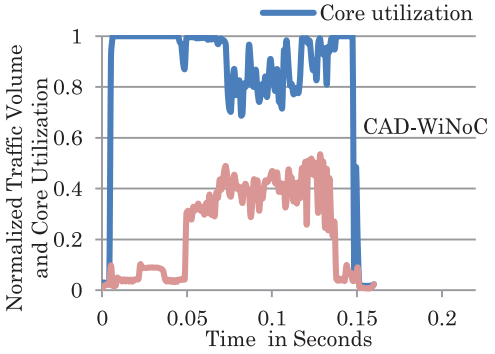
Figure 14 compares the traffic injection rates of the initial and redistribution phases. The average number of flits injected per cycle per core gives the traffic injection rate. As explained in Section 3.2, the redistribution phase involves heavy data migration. Due to their communication-intensive nature, the redistribution phases exhibit up to 4× higher injection rates when compared to corresponding initial coloring phases. Hence, as shown in Figure 13, redistribution phases achieve higher execution time improvements with improved NoC architectures when compared to the respective initial coloring phases.

The number of colors associated with each considered graph is provided in Table II. It can be noted from Figure 13 that the graphs with a larger number of colors (>80 colors) achieve a better execution-time improvement with better NoC architectures when compared to graphs with low number of colors (<20 colors). This disparity in the improvement profile can be explained as follows: The graphs with more colors also spend more time in the redistribution phase, which, as explained above, stands best to gain from the low-hop-count NoC architectures. Observing the trends in traffic volumes and CPU utilizations from Figures 15 and 16 can corroborate this. More specifically, the CPU activity drops during the redistribution phase when executed with the mesh topology. With higher average hop count, mesh NoC forces the CPUs to exhibit prolonged waits for the necessary data, leading to a drop in CPU activity. In contrast, WiNoC maintains a fairly high CPU utilization throughout the redistribution period.

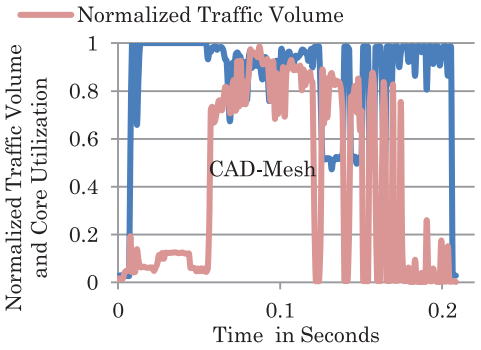
5.4. System Energy

Figures 17(a) and 17(b) shows the energy consumptions of the multicore systems executing *Grappolo* and Balanced Coloring, employing the six different NoC architectures considered in this work.

The overall energy consumption of the multicore system is the sum of the energies consumed by the cores, interconnects and network router. As stated earlier in Section 5.2, the FBF and WiFBF architectures dissipate high router energies and hence consume higher network energies when compared to SWNoC, WiNoC, and WiMesh

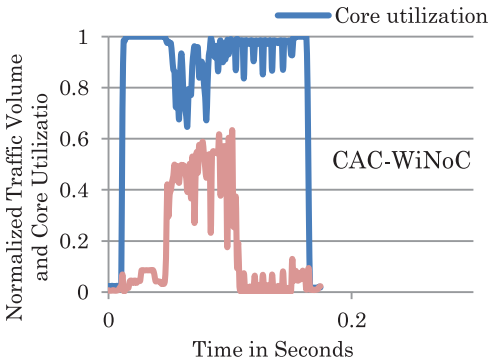


(a) CAD: Redistribution phase with WiNoC

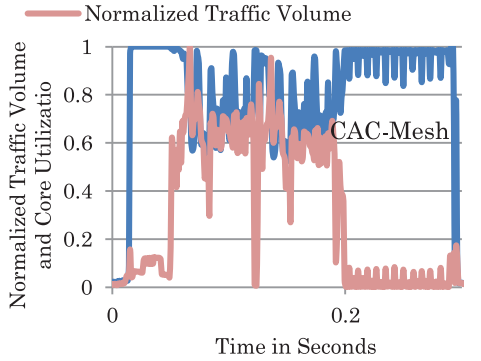


(b) CAD: Redistribution phase with mesh

Fig. 15. Balanced coloring - CAD runtime characteristics.



(a) CAC: Redistribution phase with WiNoC

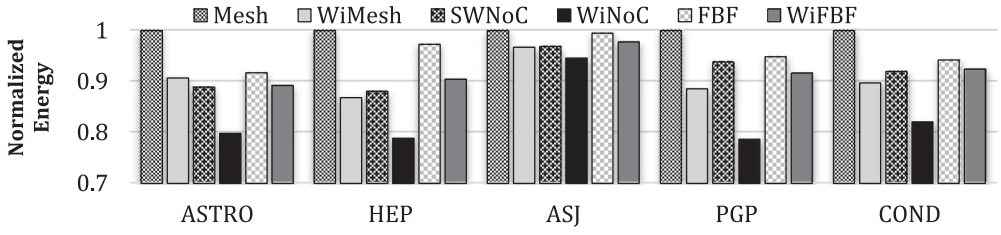


(b) CAC: Redistribution phase with mesh

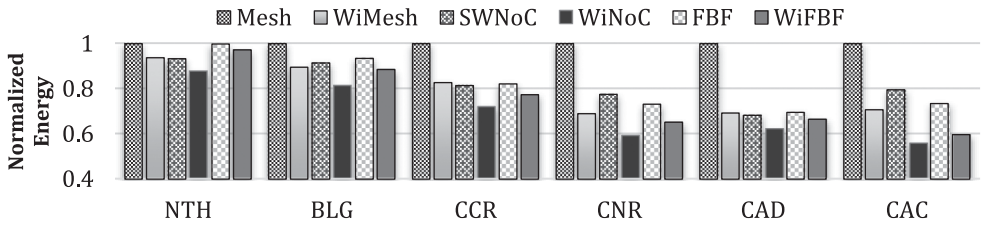
Fig. 16. Balanced coloring - CAC runtime characteristics.

architectures. The variation in network energy consumption among the mesh and small-world networks can be explained as follows: The average hop count of the wireline mesh network is higher than WiMesh and the small-world NoC architectures (Figure 9(a)). Hence, each injected message stays longer in the wireline mesh network than in the WiMesh, SWNoC and WiNoC architectures. This in turn leads to higher network energy consumptions with the wireline mesh NoC. As an example, we can consider the execution of CAD Balanced Coloring with mesh and WiNoC architectures. By comparing Figures 15(a) and 15(b), it can be seen that the average traffic volume associated with mesh is higher than the average traffic volume associated WiNoC. Moreover, compared to the wireline NoCs (wireline mesh, SWNoC and FBF) architectures, the WiMesh, WiNoC, and WiFBF NoCs use more energy-efficient wireless links for long-range communication. This enables WiMesh, WiNoC, and WiFBF to achieve a better interconnect energy consumption than their wireline counterparts. Finally, when compared to the wireline mesh, all the other architectures enable a lesser execution time, leading to improved CPU energy consumptions.

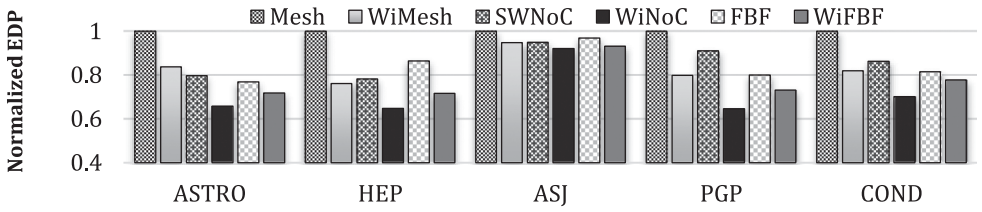
Overall, the WiNoC architecture achieves lowest energy dissipation. More specifically, WiNoC has the lowest network energy dissipation and the second lowest core energy dissipation. Based on the graphs considered, Balanced Coloring achieves a maximum of 44.3% reduction in energy consumption by using WiNoC when compared to wireline mesh. *Grappolo* achieves a maximum of 35.5% energy savings. The CPU



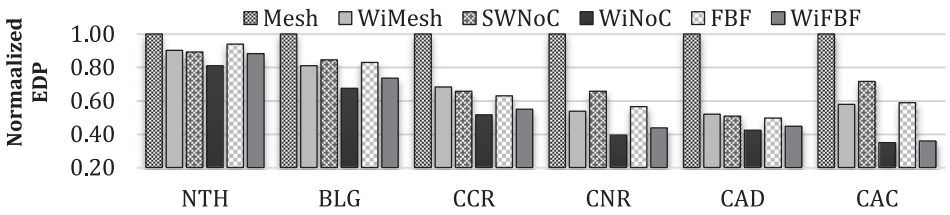
(a) *Grappolo*: Full system energy



(b) Balanced coloring: Full system energy



(c) *Grappolo*: Full system EDP



(d) Balanced coloring: Full system energy

Fig. 17. Full system energy consumption and EDP characteristics.

energy savings are more pronounced for the Balanced Coloring application due to the communication intensive nature of redistribution phase and hence Balanced Coloring application achieves higher full-system energy savings than *Grappolo*.

Figures 17(c) and (d) show the full-system EDPs of the multicore systems executing *Grappolo* and Balanced Coloring, employing the six different NoC architectures considered in this work. We use full-system execution times and full-system energy consumptions to compute full-system EDP. Due to the tradeoffs associated with reducing energy at the cost of the execution time performance, EDP is a suitable metric for analyzing the full-system performance profile. To summarize, among all the considered NoC architectures, WiNoC achieves the best full-system energy delay product. WiNoC is followed by WiFBF. Compared to the WiFBF, the WiNoC achieves a maximum of 12.5% EDP enhancement (an average of 9% EDP savings for WiNoC over

WiFBF). For the graph applications considered here, when compared to the traditional wireline mesh and the wireline FBF NoCs, the WiNoC achieves on an average of 38% and 18% EDP enhancements respectively. Moreover, it is already shown in Section 5.2.3, that the WiNoC is also a high robust architecture that can more efficiently handle the wireless link failures. From these discussions, we can conclude that the small-world network-enabled WiNoC is the most suitable NoC architecture for implementing complex large-scale graph analytics on the multicore platforms.

6. CONCLUSION

Graph analytics have become an essential part of the discovery pipeline in numerous data-driven scientific and social computing fields. However, implementing advanced graph operations on state-of-the-art multicore platforms requires significant redesign of the on-chip interconnect network topologies to mask the adverse effects of irregular data movement characteristics. In this article, we analyzed the traffic patterns generated by two state-of-the-art parallel implementations for advanced graph analytics – viz. *Grappolo* for community detection, and Balanced Coloring. The traffic generated by these applications exhibit high long-range communication patterns with traffic hotspots and are highly suitable for the implementation low hop count NoC topologies. Towards this end, in this work we explore the suitability of three different NoC topologies, mesh, small-world Network on Chip (SWNoC) and Flattened Butter-Fly (FBF), for implementing multicore graph analytics. Incorporated with long-range shortcuts, the small world and Flattened Butter-Fly topologies enable a fast data exchange among the computing cores leading to improved system performance when compared to the traditional wireline mesh. Among all the NoC architectures considered in this work, the wireless enabled small-world Network on chip (WiNoC) architecture achieves the best full system Energy Delay Product (EDP). When compared to the traditional wireline mesh and wireline FBF networks, WiNoC achieves an average of 38% and 18% improvement in EDP respectively for running graph analytics. The study presented in this article represents, to the best of our knowledge, the first detailed design-space exploration of NoC architectures for multicore graph analytics.

REFERENCES

N. Abeyratne, R. Das, Q. Li, K. Sewell, B. Giridhar, R. G. Dreslinski, D. Blaauw, and T. Mudge. 2013. Scaling toward kilo-core processors with asymmetric high-radix topologies. In *Proceedings of 19th International Symposium on High Performance Computer Architecture (HPCA2013)*. 496–507.

D. A. Bader, G. Cong, and John Feo. 2005. On the architectural requirements for efficient execution of graph algorithms. In *Proceedings of the 34th International Conference on Parallel Processing (ICPP 2005)*. 547–556.

N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Said, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. 2011. The GEM5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2, 1–7.

H. L. Bodlaendera and F. V. Fominb. 2005. Equitable colorings of bounded treewidth graphs. *Theoretical Computer Science* 349, 1, 22–30.

W. Bogaerts, M. Fiers, and P. Dumon. 2014. Design challenges in silicon photonics. *IEEE Journal of Selected Topics in Quantum Electronics* 20, 4, 1,8 (July-Aug. 2014).

J. Branch, X. Guo, A. Sugavanam, J. J. Lin, and K. K. O. 2005. Wireless communication in a flip-chip package using integrated antennas on silicon substrates. *IEEE Electronic Device Letters* 26, 2, 115–117.

M. Castro, E. Francesquini, T. M. Nguélé, and J. F. Méhaut. 2013. Analysis of computing and energy performance of multicore, NUMA, and manycore platforms for an irregular application. In *Proceedings of the 3rd Workshop on Irregular Applications: Architectures and Algorithms*.

Umit V. Çatalyürek, J. Feo, A. H. Gebremedhin, M. Halappanavar, and A. Pothen. 2012. Graph coloring algorithms for multi-core and massively multithreaded architectures. *Parallel Computing* 38, 10–11 (October 2012), 576–594.

D. Chavarría-Miranda, M. Halappanavar, and A. Kalyanaraman. 2014. Scaling graph community detection on the tilera manycore architecture. In *Proceedings of HiPC 2014*, Goa, India, 2014.

D. Chen, N. Eisley, P. Heidelberger, S. Kumar, A. Mamidala, F. Petrini, R. Senger, Y. Sugawara, R. Walkup, B. Steinmacher-Burow, and A. Choudhury. 2012. Looking under the hood of the IBM blue gene/Q network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 1–12.

W. J. Dally and C. L. Seitz. 1987. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Computer* C-36, 5 (May 1987), 547–553.

B. D. De Dinechin, D. Van Amstel, M. Poulhies, and G. Lager. 2014.Time-critical computing on a single-chip massively parallel processor. In *Proceedings of IEEE DATE*. 1–6.

S. Deb, A. Ganguly, P. P. Pande, B. Belzer, D. Heo. 2012. Wireless NoC as interconnection backbone for multicore chips: Promises and challenges. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 2, 2, 228–239.

S. Deb, K. Chang, Yu Xinmin, S. P. Sah, M. Cosic, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo. 2013. Design of an energy efficient CMOS compatible NoC architecture with millimeter-wave wireless interconnects. *IEEE Transactions on Computers* 62, 12, 2382–2396.

DIMACS10. 2016. The 10th DIMACS implementation challenge – Graph partioning and clustering. URL: http://www.cc.gatech.edu/dimacs10/ (Last date accessed: May 2016).

K. Duraisamy, R. G. Kim, and P. P. Pande. 2015. Enhancing performance of wireless NoCs with distributed MAC protocols. In *Proceeedings of ISQED*. 2015.

D. Ediger. 2013. *Analyzing Hybrid Architectures for Massively Parallel Graph Analysis*. Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, Ga., (May 2013).

S. Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3, 75–174.

E. Francesquini, M. Castro, P. H. Penna, F. Dupros, H. C. Freitas, P. O. Navaux, and J. F. Méhaut. 2015. On the energy efficiency and performance of irregular application executions on multicore, NUMA and manycore platforms. *Journal of Parallel and Distributed Computing* 76, 32–48.

M. Frasca, K. Madduri, and P. Raghavan. 2012. NUMA-aware graph mining techniques for performance and energy efficiency. In *Proceedings of IEEE International Conference on High Performance Computing*. Networking, Storage and Analysis (SC), 1–11.

H. Furmanczyk. 2004. Equitable coloring of graphs. In *Graph Colorings*, M. Kubale (Ed.). Contemporary Mathematics, Vol. 352. American Mathematical Society, Providence, Rhode Island, 35–53.

L. Gwennup. 2011. Adapteva: More flops, less watts: Epiphany offers floating-point accelerator for mobile processors. *Microprocess. Rep* 2, 1–5.

T. R. Jensen and B. Toft. 1995. *Graph Coloring Problems*. Wiley Series in Discrete Mathematics and Optimization, Wiley Interscience, New York.

M. T. Jones and P. E. Plassmann. 1993. A parallel graph coloring heuristic. *SIAM Journal on Scientific Computing* 14, 3, 654–669.

J. Kim, J. Balfour, and W. J. Dally. 2007. Flattened Butter-Fly: A cost-efficient topology for high-radix networks. *IEEE Computer Architecture Letters* 6, 2, 37–40.

T. Krishna, A. Kumar, P. Chiang, M. Erez, and L. Peh. 2008. NoC with near-ideal express virtual channels using global-line communication. In *Proceedings of the 16th IEEE Symposium on High Performance Interconnects (HOTI'08)*. 11–20, 26–28.

T. Krishna, C. O. Chen, S. Park, W. C. Kwon, S. Subramanian, A. P. Chandrakasan, and L. Peh. 2013. Single-cycle multihop asynchronous repeated traversal: A smart future for reconfigurable on-chip networks. *IEEE Computer* 10, 48–55.

T. Krishna, C. O. Chen, W. C. Kwon, and L. Peh. 2014. Smart:single-cycle multihop traversals over a shared network on chip. *IEEE Micro* 34, 3, 43–56.

A. Kumar, L. Peh, P. Kundu, and N. K. Jha. 2008. Toward ideal on-chip communication using express virtual channels. *IEEE Micro* 28, 1, 80–90.

F. T. Leighton. 1979. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards* 84, 6, 489–506.

S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. 2009. McPAT:An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture.* 469–480.

J. J. Lin, H. Wu, Y. Su, L. Gao, A. Sugavanam, J. E. Brewer, and K. K. O. 2007. Communication using antennas fabricated in silicon integrated circuits. *IEEE Journal of Solid-State Circuits* 42, 8, 1678–1687.

H. Lu, M. Halappanavar, and A. Kalyanaraman. 2015b. Parallel heuristics for scalable community detection. *Parallel Computing* 47, 19–37.

H. Lu, M. Halappanavar, D. Chavarria-Miranda, A. Gebremedhin, and A. Kalyanaraman. 2015a. Balanced coloring for parallel computing applications, In *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 25–29, Hyderabad, India.

R. Marculescu, U. Y. Ogras, Peh Li-Shiuan, N. E. Jerger, and Y. Hoskote. 2009. Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 1, 3–21.

M. E. J. Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 103, 23, 8577–8582.

U. Y. Ogras and R. Marculescu. 2006. It's a small world after all: NoC performance optimization via long-range link insertion. *IEEE Trans. Very Large Scale Integration Systems.* 14, 7, 693–706.

P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. 2005. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE Transactions on Computers*. 54, 8, 1025–1040.

T. Petermann and P. De Los Rios. 2005. Spatial small-world networks: A wiring cost perspective. *arXiv: Condmat/0501420v2*.

J. E. Riedy, H. Meyerhenke, D. Ediger, and D. A. Bader. 2012. Parallel community detection for massive graphs. In *Parallel Processing and Applied Mathematics*. Springer, Berlin, 286–296.

Y. Saad. 2003. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

E. E. Schadt, M. Linderman, J. Sorenson, L. Lee, and G. P. Nolan. 2010. Computational solutions to large-scale data management and analysis. *Journal of Nature Reviews Genetics* 11, 9, 647–657.

E. Seok and K. K. O. 2005. Design rules for improving predictability of on-chip antenna characteristics in the presence of other metal structures. In *Proceedings of IEEE International Interconnect Technology Conference*. 6–8, 120–122.

K. Sewell, R. G. Dreslinski, T. Manville, S. Satpathy, N. Pinckney, G. Blake, M. Cieslak, R. Das, T. F. Wenisch, D. Sylvester, D. Blaauw, and T. Mudge. 2012. Swizzle-switch networks for many-core systems. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 2, 2, 278–294.

C. L. Staudt and M. Meyerhenke. 2013. Engineering high-performance community detection heuristics for massive graphs. In *Proceedings of 42nd International Conference on Parallel Processing (ICPP)*. 180–189.

Tilera Corporation. 2015. *TILE-Gx72 Processor Product Brief*. http://www.tilera.com/files/drim__TILE-Gx8072_PB041-04_WEB_7683.pdf (Last Accessed: May. 2016).

D. J. Watts and S. H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Letters to Nature*. 393.6684, 440–442.

P. Wettin, R. Kim, J. Murray, Yu Xinmin, P. P. Pande, A. Ganguly, and D. Heoamlan. 2014. Design-space exploration for wireless NoCs incorporating irregular network routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 11, (Nov. 2014), 1732–1745.

B. Wu, Y. Dong, Q. Ke, and Y. Cai. 2011. A parallel computing model for large graph mining with MapReduce. In *Proceedings of 7th International Conference on Natural Computation (ICNC)*. 43–47.

Y. P. Zhang, Z. M. Chen, and M. Sun. 2007. Propagation mechanisms of radio waves over intra-chip channels with integrated antennas: Frequency-domain measurements and time-domain analysis. *IEEE Transactions on Antennas and Propagation* 55, 10, 2900–2906.

Received December 2015; revised May 2016; accepted May 2016