International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 6, No.2 (Mar-2017)
http://dx.doi.org/10.12785/ijcds/060201

Clustered Networks-on-Chip: Simulation and Performance Evaluation

Ahmed S. Hassan[1], Ahmed A. Morgan[2] and M. Watheq El-Kharashi[1,3]

[1]*Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt*
[2]*Department of Computer Engineering, Cairo University, Giza, Egypt*
[3]*Electrical and Computer Engineering Department, University of Victoria, Victoria, Canada*

Received 11 Dec. 2016, Revised 8 Jan. 2017, Accepted 5 Feb. 2017, Published 1 March 2017

Abstract: In many-core Networks-on-Chip (NoC) systems, two topics have recently been researched; NoC clustering and NoC simulation. NoC clustering investigates grouping processing elements (PEs) based on common characteristics between them, like spatial locality or communication patterns. Research showed that NoC clustering achieved better performance and load balancing. Furthermore, it allows scalability of the NoC grid. The other topic, NoC simulation, provides tools for early evaluation of NoC systems. This allowed easy investigation of NoC systems with large number of PEs. Plenty of research has been done in both topics separately. In this paper, we try to fill the gap. We extend an existing NoC simulation tool by adding support for simulating NoC clusters and inter-NoC traffic. The presented NoC clustering simulator supports the modular design technique, which in turn offers the flexibility in configuring cluster parameters. Examples of configurable parameters that are supported by our modified simulator are: adding interconnection topologies to configure how the NoC entities are connected with each other, adding a cluster manager that maps tasks to NoC entities based on the inter-NoC communication pattern, adding latency factors of cluster links in the case of inter-NoC traffic, and adapting the routing algorithm for both inter-cluster and intra-cluster traffic. A clustered NoC simulation case study shows the effectiveness of the modifications made to the simulator. For example, a certain NoC setup is simulated twice, initially non-clustered then clustered, using the added features. Collected data shows that the average clustered NoC routers' energy consumption is 50% lower than the consumption in the non-clustered case.

Keywords: Clustering, Inter-NoC communication, NoC, NoCTweak

1. Introduction

Use of many-core System-on-Chip (SoC), and especially Networks-on-Chip (NoC) based platforms, have increased dramatically. These platforms are loaded with lots of processing elements (PEs) and other peripherals. As the number of connected PEs increases, their communication cost increases, in terms of hop count, and packets are more prone to stalls and contention. Depending on task mapping and routing protocols, the generated traffic could result in high communication density between some PEs and over certain links, in which PEs maybe close to one another or in different neighborhoods.

In many-core NoC systems, two topics have been researched recently; NoC clustering and NoC simulation. NoC clustering investigates grouping PEs based on common characteristics, like spatial locality or communication pattern, to form sub-NoC entities, then those sub-NoC entities are placed in a bigger grid according to a given mapping criteria. Research showed that NoC clustering achieved better performance and load balancing. Furthermore, it allows scalability of the NoC grid. The other topic, NoC simulation, provides tools for early evaluation of many-core NoC systems. This allows easy investigating of NoC systems with large number of PEs. The system designer has to explore different NoC topologies and routing techniques, along with different traffic patterns, to reach the optimal performance and energy dissipation [1], or trade-off between area and average delay [2]. This early design space exploration comes with high cost, in both time and resources.

There are two types of NoC simulators, systems-level and circuit-level. In system-level simulation, the NoC system design is evaluated as a whole. Parameters, like routing algorithm efficiency and topology, are evaluated in terms of communication latency and throughput [3]–[10]. System modeling can provide estimates for system parameters, like buffer and queue sizes [11]. As for circuit-

E-mail address: ahmedsayed.elaraby@gmail.com, ahmorgan@eng.cu.edu.eg, watheq.elkharashi@eng.asu.edu.eg

level simulation, technology parameters are taken into consideration, like critical path delays and temperature variation [12]. For early design space exploration, system-level simulators are more suitable.

Many literatures have investigated system-level NoC simulation for early design space exploration. Noxim [3] and NoCTweak [4] are notable examples. Most of the NoC simulators focus on evaluating metrics, like throughput and energy dissipation on the NoC platform level. They further allow customizing the traffic communication pattern and rate, the routing algorithm, and the topology. NoC clustering has spread out and different clustering techniques have been investigated to provide both optimal performance and energy dissipation [13]–[17]. However, those NoC clustering techniques are not included in the mainstream NoC simulators. In our work, we addressed this problem by modifying one of those simulators in order to support NoC clustering simulation. We modified an existing NoC simulation tool: NoCTweak. The following NoC clustering simulation-related issues are investigated in our work: NoC clusters platforms, traffic routing, and cluster management.

The above NoC simulators focused on simulating a single NoC system, ignoring the scalability of the NoC platform, which can be achieved by clustering multiple NoC systems. Consequently, the interaction between NoC clusters is not considered. In this paper, we fill this open research gap by providing a tool to simulate NoC clusters. We decided to select a NoC simulation tool and extend it to support simulation traffic between multiple NoC systems.

Out of the existing NoC simulators, we selected NoCTweak [4] to modify for clustered NoC simulation. *NoCTweak* large configuration options, especially the link length, motivated us to select it for extension.

The following features are added to the original *NoCTweak*:

1) Cluster Interconnection Topologies: How NoC entities are connected with each other.
2) Latency Model: How latency is modeled and calculated in the case of inter-NoC traffic.
3) Flit Routing: How flits are routed across the cluster. There are two options here:
   a) Same algorithm for inter- and intra-cluster routing
   b) Different algorithms for inter- and intra-cluster routing
4) Cluster Manager: How to map tasks to NoC entities, based on the inter-NoC communication pattern.

The contributions of this paper are:

1) Introducing the concept of inter-NoC traffic simulation.
2) Extending a NoC simulator, *NoCTweak*, to support inter-NoC communication while evaluating NoC performance.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 provides background on NoC clustering. Section 4 reviews the original *NoCTweak* simulator. Section 5 presents the new feature added in *NoCTweak* to support simulating NoC clusters. Section 6 presents a clustered NoC simulation case study. The paper is concluded in Section 7.

2. NoC Simulation Related Work

A handful of NoC simulators have been developed over recent years. The main focus of NoC simulators was to select and customize the NoC parameters, like the routing algorithms, the communication pattern, the topology, the router microarchitecture, and the traffic characteristics. In the following sections we explore selected NoC simulators.

A. Noxim

Noxim was developed by Palesi et al., and implemented in SystemC [3]. Noxim provides a variety of options to simulate 2-D NoC. The user can customize simulations with different NoC sizes, router buffer sizes, flit sizes, routing algorithms, and flit injection rates. The simulation evaluates the NoC in terms of throughput, delay, and power consumption. It also has the option to give detailed statistics for each communication link per destination node.

B. BookSim

Jiang et al. developed BookSim, which allows configuring topology, buffer size, routing algorithm, and router micro-architecture [5]. Figure 1 shows BookSim high-level architecture. It incorporates a top-level traffic generator, which wraps around the network under simulation.

C. GARNET

Most of the traditional NoC simulators are not full-system simulators. They focus on simulating the NoC interconnect behavior, producing results about evaluating throughput and latency of the interconnect. Agarwal et al. implemented GARNET as a full-system NoC simulator, which enables the evaluation of components, like caches and memory controller, along with NoC topology, router microarchitecture, and routing algorithms [6].

D. NoCTweak

Tran et al. implemented *NoCTweak*, which allows a wide range of configurations to be applied on the NoC

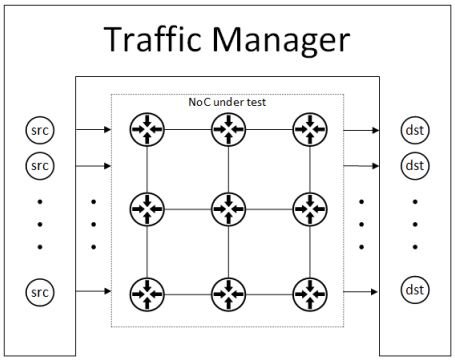


Figure 1. Top-level block diagram of BookSim [5].

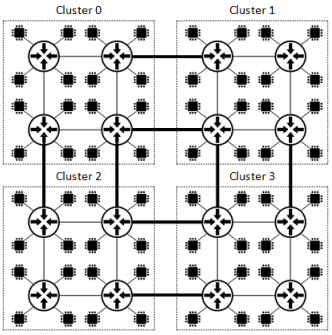


Figure 2. C-NOC cluster architecture [13].

platform under simulation [4]. *NoCTweak* has many configuration options that are not found in other simulators, e.g., completely controlling traffic patterns, number of flits per packet, pipeline stages, switch and virtual channel allocation policies, and inter-router link length.

3. NoC Clustering Background

With the steadily growing number of PEs in NoC platforms, it became necessity to employ clustering techniques that group related and heavily-communicated PEs close to one another. Employed clustering technique should reduce hop distance between communicating clusters as much as possible. NoC clustering provides means for load balancing, by task distribution and relocation, and resource borrowing among clusters. Some NoC clustering techniques require a PE to act as a manager for the cluster [17].

Clustering can be static, where the shape and size of the cluster does not change at runtime, or dynamic, where the cluster size may change during runtime. Dynamic clustering can be done by borrowing PEs from neighboring clusters [16], [17], or by programmable-switch fabric [18]. Cluster management can be done by a specific node, i.e. the cluster manager, or handled entirely in the router logic and routing algorithm.

A. Static Clustering

1) C-NOC

A basic NoC clustering technique is introduced in Clustered NoC (C-NOC) [13]. In C-NOC, the traditional router with four ports has been extended to have other ports that connect to other clusters. Figure 2 shows a 2×2 C-NOC cluster connection, and each cluster is a 2×2 NoC.

2) CBHR

Saravanakumar et al. introduced Cluster Based Hierarchical Routing (CBHR) by logically dividing the NoC nodes into smaller clusters [14]. To do this logical clustering, the traffic is routed internally within a cluster till it reaches a boundary router and then the traffic is routed between clusters. Inter- and intra-cluster routing algorithm can be different. Choosing internal or global routing function depends on the header flit format, which differs between a packet sent to inter- or intra-cluster destination. Figure 3 shows how routing works.

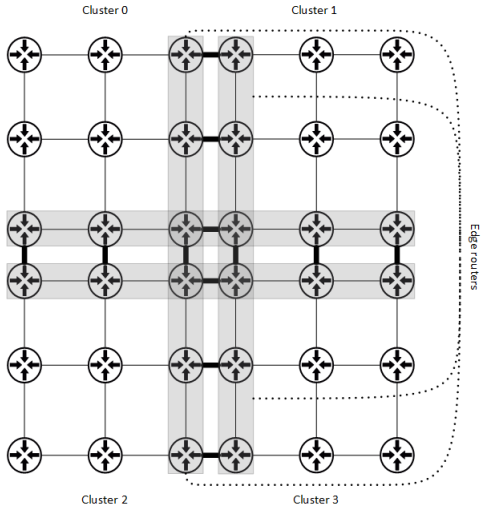
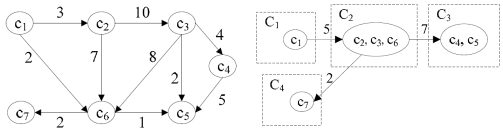


Figure 3. Routing across different clusters in CBHR [14]. Routing algorithms for edge routers can be different from algorithms for used in other routers.

3) Cluster-TG

Ge et al. introduced Cluster-based NoC Topology Generation (Cluster-TG) [15]. Cluster-TG generates irregular topology for a given application, by grouping nodes according to their communication bandwidth. Figure 4 shows an example of Cluster-TG clustering. Cluster-TG clustering technique can minimize power consumption and area cost.



(a) Communication graph (b) Cluster-TG result

Figure 4. Cluster-TG core clustering) [15].

4) CSA

Lu et al. introduced Clustering-based Simulated Annealing (CSA) algorithm for core-to-node mapping, knowing the communication demands of the application [19].

B. Dynamic Clustering

1) Decentralized Agent Based Re-clustering

Cui et al. introduced a task mapping algorithm, to support scalability [16]. This is achieved by changing NoC cluster size during runtime, depending on the system load and communication pattern. The proposed technique initially groups processing cores into clusters, and dynamically changes the clusters size according to task mapping, as depicted in Figure 5. If a task is mapped to a cluster, which does not have sufficient resources to fulfill this task, then this cluster borrows resources from its neighbors.

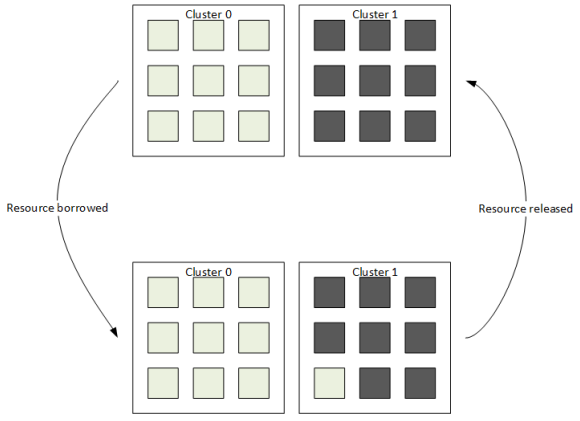


Figure 5. Runtime dynamic re-clustering [16].

2) Distributed Resource Management

Castilhos et al. introduced a distributed resource management in NoC [17]. This method relies on a cluster manager that controls the cluster, in terms of dynamic task mapping, task monitoring, and deadlines verification. Depending on the task requirements, a cluster may request to borrow resources from other clusters, also tasks can be migrated between clusters. Figure 6 depicts the NoC cluster architecture introduced in [17]. The NoC is divided into multiple regions, each has one Local Master (LMP) and different Slaves (SP). The NoC platform has one Global Master (GMP), which receives requests for task allocation and sends the requested task accordingly.

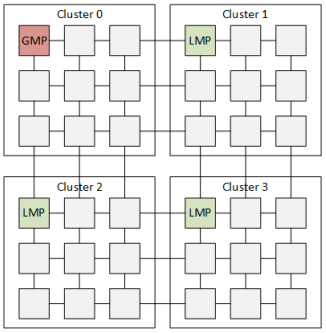


Figure 6. Global and local cluster managers in Clustered Architecture [17].

4. The Original NoCTweak Simulator

NoCTweak is an open-source NoC simulator that is based on SystemC [4]. It is used for exploring the performance and energy dissipation of different NoC platforms.

A. Configuration Parameters

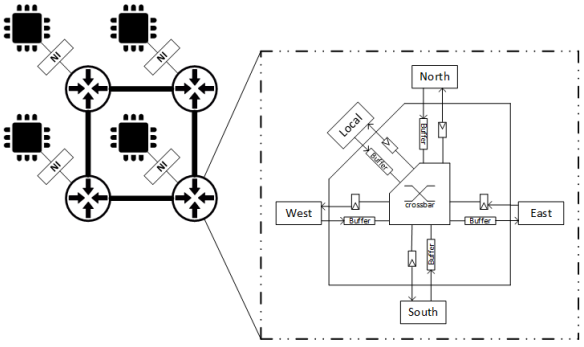


Figure 7. Generic 2-D NoC platform simulated by NoCTweak [4].

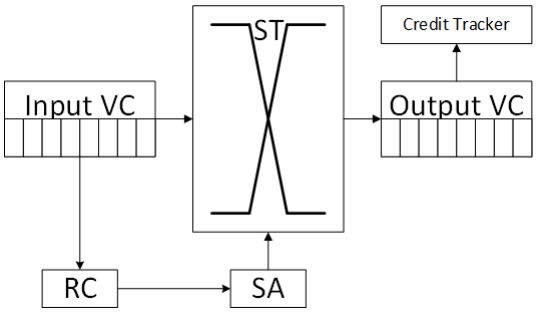
NoCTweak is highly parametrized, allowing setting up and simulating a board range of NoC configurations, such as router type, network size, buffer size, routing algorithm, arbitration policy, pipeline stages, supply voltage, clock frequency, traffic pattern, packet length, injection rate, simulation and warm-up times [4].

Original *NoCTweak* simulates only 2-D mesh NoC platforms, as shown in Figure 7, where every PE is consisted of a processing core and a network interface (NI). Each PE node is linked to a router. Each router is connected with its neighbors in the mesh.

B. Router Models

NoCTweak simulates multiple router models, like Wormhole, Virtual-channel, RoShaQ [20], bufferless, and circuit-switched routers.

The *NoCTweak* default router is the Wormhole router. Figure 8 shows the implementation of the wormhole router, and Figure 8a shows a simplified diagram of the Wormhole router architecture used in *NoCTweak*. Router pipeline configuration is flexible, it can be between one and five pipeline stages. Figure 8b shows the different pipeline stages, namely *Buffer Write*, *Routing Computation*, *Switch Allocation*, *Switch Traverse*, and *Link Traverse*.



(a) Component diagram.

(b) Pipeline stages.

Figure 8. Implementation of NoCTweak Wormhole router.

C. Statistic Outputs

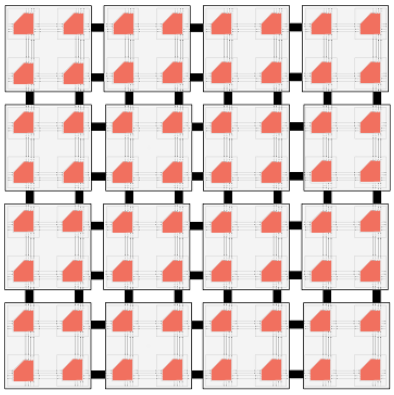
Simulation statistics are generated for network latency, network throughput, and average energy dissipated by each router. Activities of circuit components of all routers in the network are tracked and recorded as well.

5. NoC Clustering Simulation

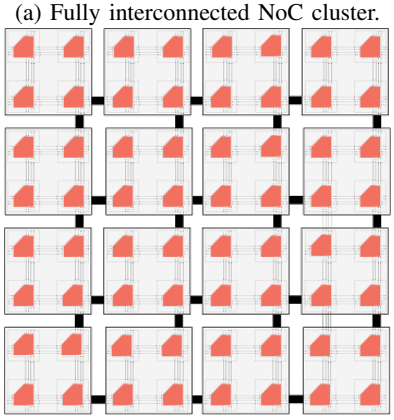
In this section, we present the changes we applied on *NoCTweak* in order to support NoC clustering simulation. We maintained the regular 2-D NoC topology, as the original simulator, and added new command-line options for clustering. The following terminology is used: *NoC cluster* is the group of PEs and *NoC grid* is the group of interconnected NoC clusters.

A. Cluster Interconnection Topologies

Our simulator supports two types of cluster interconnections, as shown in Figure 9, which depicts a 4×4 NoC cluster grid with each cluster has 2×2 PE. In the first type, all of the edge routers of a cluster are interconnected with neighboring clusters, as shown in Figure 9a. The second type has only one link between a cluster and its neighbors, as in shown Figure 9b. For different PEs to be able to communicate, the location of the cluster is added in the source and destination addresses.



(a) Fully interconnected NoC cluster.



(b) NoC cluster with one link per neighbor.

Figure 9. Types of NoC cluster interconnection.

B. Latency Model

Latency in the *NoCTweak* is calculated by getting the difference in time stamps between packet head injection (from the source PE) and tail reception (at the destination PE). Part of this delay is the Link Traverse (LT) delay. The inter-router link length is used in calculating the LT delay, which in turn affects the pipeline time and the operating frequency. The LT delay is calculated only at software initialization.

In the case of clustered NoC, the inter-cluster links are much longer than local intra-cluster links. Therefore, we have added on that a weight on the inter-cluster links in order to enhance the accuracy of the calculation. This weight can be considered as the ratio of the inter-cluster link length to that of the intra-cluster one.

For example, lets consider Ethernet as the inter-cluster link. If a packet would traverse intra-cluster link in 1 cycle, and the same packet would traverse the inter-cluster Ethernet link in 100 cycles, the weight in this case is 100.

C. Flit Routing

At initialization, each PE in the system is given a unique ID. This ID is based on the coordinates of its cluster and the coordinates within the cluster, given by this equation:

$$ID = (x + (y \times x_{dim})) + ((cluster_x + (cluster_y \times cluster_{Xdim})) \times (x_{dim} \times y_{dim})) \quad (1)$$

where x and y are the PE X and Y location within its cluster, x_{dim} and y_{dim} are the single NoC cluster dimensions, $cluster_x$ and $cluster_y$ are the X and Y location of the cluster within the cluster grid, and $cluster_{Xdim}$ is the X dimension of the NoC cluster grid. If a PE wants to communicate with another PE, it decodes both the source PE and the destination PE IDs and add the IDs to the flit.

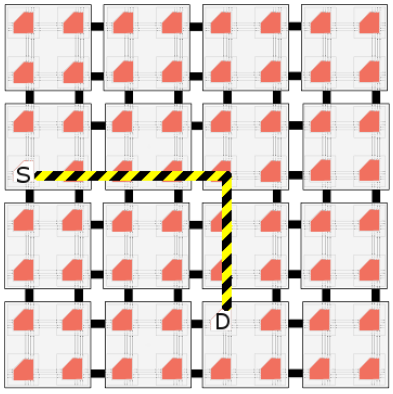
There are two ways to apply traffic routing algorithms. Either to apply the same selected routing algorithm on the whole system, or select different routing algorithms for inter-cluster traffic than intra-cluster traffic. For both ways, we used the routing algorithms provided by *NoCTweak* to set the routing algorithm for inter- and intra-cluster traffic. *NoCTweak* provides these routing algorithms: XY, Negative-First minimal adaptive, West-First minimal adaptive, North-Last minimal adaptive, Odd-Even minimal adaptive, and lookup-table based routing.

1) Same Algorithm for Inter- and Intra-cluster Routing

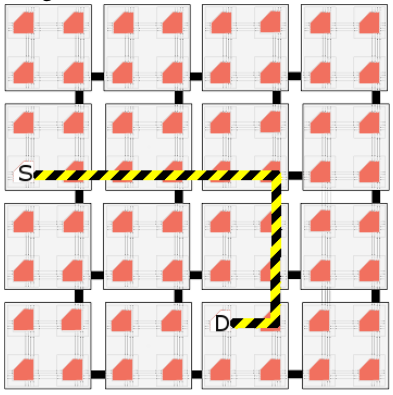
Packets will be routed between clusters and within each cluster. Depending on the cluster interconnection type, which was discussed in section 5-A, the meaning of this feature differs.

If the clusters are fully interconnected, then the routing algorithm is applied on the system as a whole. The system is treated as a single cluster of size $(x_{dim} \times cluster_{Xdim}) \times (y_{dim} \times cluster_{Ydim})$, where $cluster_{Ydim}$ is the Y dimension of the NoC cluster grid. This is not different from the original *NoCTweak* simulation, except it is applied on a larger scale and the latency model is different due to the inter-cluster links. Figure 10a shows how XY routing algorithm is applied on a fully interconnected NoC cluster grid.

For the case of one interconnection link between clusters, the algorithm is applied within each cluster to reach the router that interconnects the next cluster according to the routing algorithm. Figure 10b shows an example of XY routing. In this example, flits are routed within the source cluster till it reaches the cluster interconnection link, then flits are routed through the grid's x-dimension. Thereafter, flits are routed within that cluster to reach the interconnecting link. They are then routed through the grid's y-dimension. Finally, they are routed within the destination cluster to reach the destination PE.



(a) XY routing in case of cluster with full interconnection.



(b) Example of XY routing in case of cluster with one link per neighbor.

Figure 10. XY routing for NoC cluster interconnection.

2) Different Algorithms for Inter- and Intra-cluster Routing

When different algorithms are selected for inter- and intra-cluster routing, the simulator behaves as the case of clustering with one link between a cluster and its neighbor. However, the routing algorithm within the cluster is different from the one for inter-cluster routing. The routers execute different routing algorithms based on their position, i.e. if they are cluster-internal or cluster-interface routers.

D. Cluster Manager

In our simulator, we adapted a centralized cluster management technique, where a global system manager is

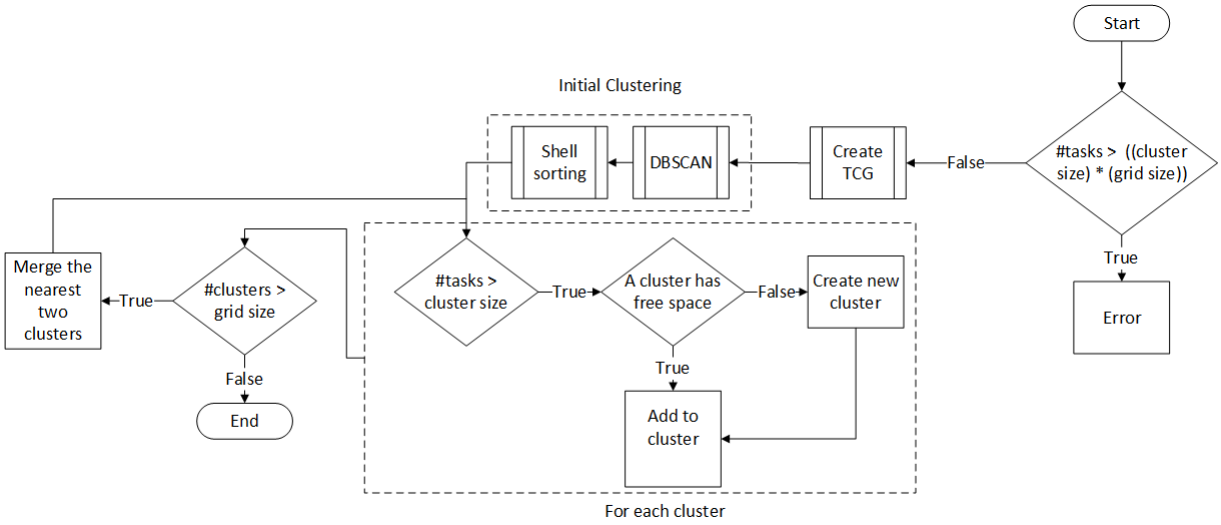


Figure 11. Algorithm for assigning tasks to NoC clusters.

TABLE I. Cluster configuration options

Parameter	Description	Default value
cdimx	X dimension length of the NoC grid	1
cdimy	Y dimension length of the NoC grid	1
cintercon	Cluster interconnection type FULL : All edge links are interconnected MIN : Only one link interconnects a neighbor cluster	FULL
crouting	Routing algorithm for inter-cluster packet routing	Algorithm selected for NoCTweak
clinkfactor	Delay factor for inter-cluster links	1.0
cmanager	Enable the use of cluster manager	

responsible for mapping tasks to clusters based on the task communication graph (TCG). The simulator is given a TCG file, describing the number of tasks to map and the communication between tasks.

The algorithm for mapping the tasks is implemented, as shown in Figure11. The cluster manager takes a TCG and does an initial clustering of tasks. Then, it iterates over the clusters, merging and splitting clusters. The mapping stops when all tasks are mapped to clusters, and no cluster is assigned a number of tasks exceeding its capacity.

For sake of simplicity, we based our clustering algorithm on DBSCAN [21]. This algorithm has an overall average runtime complexity of O(n log n). Another NoC-specific algorithm can be used to obtain optimum performance, either power optimization [22], area optimization [23], network reliability [24], genetic algorithm-based technique [25], or by using multi-objective task mapping [26]. Mapping a resultant task cluster to a location within the cluster grid is done according to Shell sorting.

For task mapping within a cluster, we relied on the original *NoCTweak* task mapping logic.

E. Cluster Configuration Options

In our extension, we provide configuration options for NoC grid size, cluster interconnection type, inter-cluster routing algorithm, inter-cluster links delay factor, and using cluster manager. Intra-cluster routing is configured using the original *NoCTweak* option "-routing". The description of newly presented options is listed in Table I. An example of the use of these configurations is given in Listing 1.

Listing 1. NoCTweak simulation configurations

```
dimx 10 dimy 10 routing xy outsel
    highercredit bsize 8 sa rr
    length 10 fir 0.50
    -cdimx 4 -cdimy 4 -cintercon FULL -crouting xy
```

The aforementioned configurations are passed along with the original *NoCTweak* options. Our current implementation supports taking only one set of cluster configuration options, which is configured using the original *NoCTweak*

options and applies this set on all of the clusters in the NoC grid.

6. Clustered NoC Simulation Case Study

In this section, we discuss the results obtained when using our simulator to evaluate different NoC cluster designs to validate the versatility of clustering parameters and assess simulation time versus NoC cluster and grid sizes. Listing.1 shows the configuration options that were used. The options in italic format are the options added to configure the NoC cluster grid, according to Table I. Simulations are executed till 100,000 packets are exchanged. Simulations where done on a Windows 10 machine with a 2.2GHz Intel i7 Quadcore processor and 6 GB RAM.

A. Simulation Time

We simulated non-clustered NoC, which is the original *NoCTweak*, and cluster grid of size 2×2, 4×2, and 4×4. For each cluster grid, cluster sizes of 10×10, 8×8 and 4×4 are simulated. Random tasks are mapped into the NoC, 100 tasks are mapped to cluster of size 10×10, 64 tasks are mapped to cluster size 8×8, and 16 tasks are mapped to cluster size of 4×4. For the *NoCTweak* configurations, we have used the default option in most of the configurations. To test the system under different traffic loads, we varied the Flit Injection Rate (FIR) from 0.2 to 0.5.

Figure 14 shows the time taken to run the simulation with different cluster and grid sizes. For example, simulating non-clustered 10×10 NoC took about 211 seconds, and simulating a grid of size 4×4, which contains 16 clusters of 10×10 PE, took about 4500 seconds.

On average, the clustering simulation and task mapping, add about 0.3 overhead over the time it takes to simulate a single NoC system, multiplied by the number of NoC clusters in the grid. This time increase is due to the cluster manager overhead, and the logic for the modified latency model. Considering the benefits of using clustering with large scale NoC, the overhead in simulation time is not much.

B. Packet Latency

Figure 12 shows the result obtained by running different FIR on different cluster sizes with different grid sizes. Each sub-figure represents a grid size, while the legend within each sub-figure represents the cluster size. In the non-clustered case, the packet latency increases with both the FIR and the cluster size. This makes sense as with increasing the FIR, more flits are stalled in the routers' buffers. Furthermore, with the increase in the cluster size, flits travels more hobs.

As we increase the grid size, keeping the same number of tasks, the average hop count a packet traverses is increased, which in turn increases the packet latency. Our simulation shows a degraded performance between the non-clustered case and the other clustered ones. Packet latency increases, with an average of 58%, as we increase the grid dimension.

Degradation in packet latency may be contributed to the clustering algorithm. Clustering techniques, other than DBSCAN, would result in lower latency increase. However, these clustering techniques have higher execution time complexity and their implementation are left for future work.

Similarly, we used *NoCTweak* default mapping function to map tasks onto PEs within the cluster. However, other mapping functions would result in better latency results. For example, task mapping within each cluster should consider mapping tasks with high inter-cluster traffic to edge PEs. Task mapping should also consider the cluster interconnection type. Moreover, task mapping should consider whether the cluster is fully interconnected or has one link per neighbor, in order to minimize intra-cluster routing overhead.

C. Router Power and Energy

Figure 13 shows simulation results for power and energy. Results were obtained by running the simulation with a 0.5 FIR. Figure 13a shows that clustering helped lowering the average power consumption per router with about 50%. The main driver behind this improvement in power consumption is that the buffering, arbitrating, and credit tracking activities in the router have been decreased. Task mapping, resulted from clustering, generated traffic that spread across most of the routers evenly. This lowered the average router power consumption.

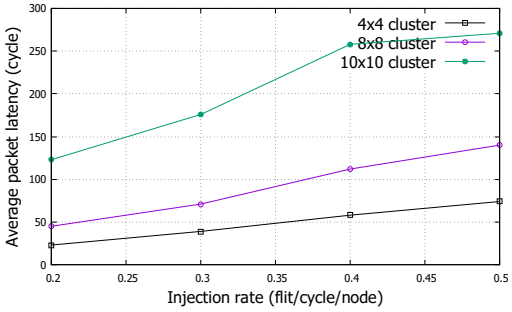
Figure 13b shows the average energy per packet per router. Energy is calculated by as:

$$AveragePacketEnergy = \frac{(AveragePower)/(ClockFrequency)}{(TotalNumberOfPackets)} \quad (2)$$

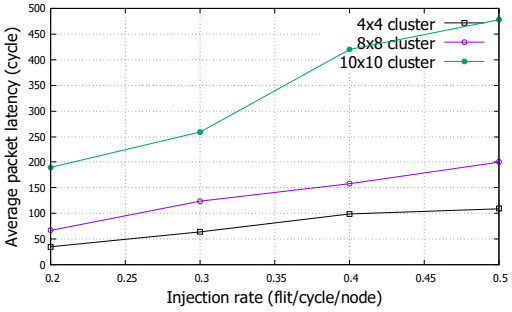
According to (2), as the average router power decreases in clustering an NoC, while maintaining the same total number of packets, the average packet energy also decreases compared to the non-clustered case.

7. Conclusion And Future Work

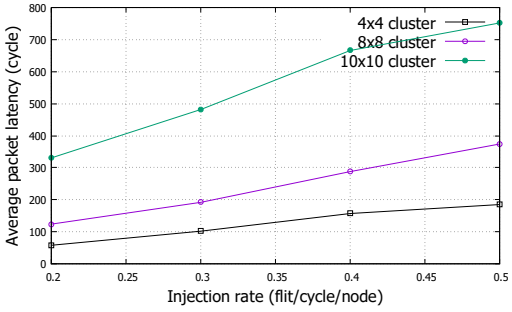
In this paper, we presented an extension to *NoCTweak* NoC simulator to support simulating NoC clustering. The new simulator features include grouping multiple NoC clusters in a grid and configuring the cluster interconnection type. Our extension allows having different routing algorithms for inter and intra-cluster traffic. Also, it allows



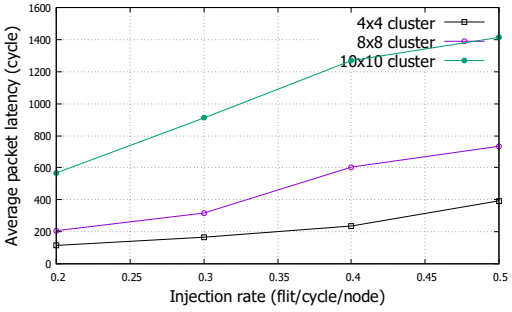
(a) Non-Clustered.



(b) 2×2 grid.

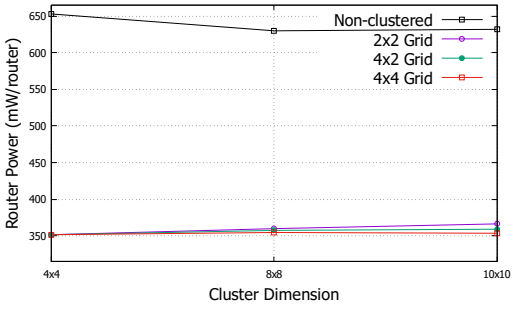


(c) 4×2 grid.

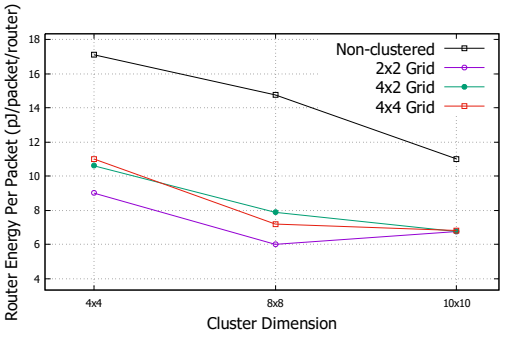


(d) 4×4 grid.

Figure 12. Average packet latency versus flit injection rate.



(a) Average router power versus cluster size.



(b) Average router energy per packet versus cluster size.

Figure 13. Average router power and energy.

adding delay factor on the inter-cluster links and adding a cluster manager.

In the future, we aim to verify the simulation results against actual implementation of NoC clusters on an FPGA. We further want to extend the simulator to support dynamic clustering, dynamic scaling, resource borrowing, and the ability to configure different NoC clusters with different configurations. As for clustering technique, we aim at supporting the techniques mentioned in [13]–[17], [19], [27]–[30], and allowing users to specify their own custom technique as well.

ACKNOWLEDGMENT

This paper is a significant extension and update of a paper that appeared in the proceedings of the Workshop on Design and Performance of Networks on Chip (DP-NoC 2016) in conjunction with the The 11th International Conference on Future Networks and Communications (FNC 2016) [31].

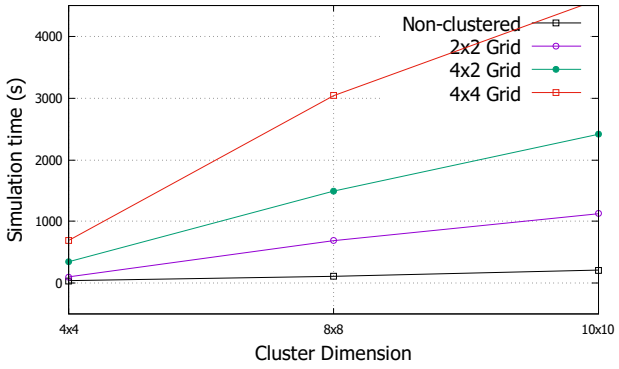


Figure 14. Cluster simulation time for different cluster and NoC sizes.

References

[1] F. Gebali, H. Elmiligi, and M. W. El-Kharashi, *Networks-on-chips: theory and practice*. CRC press, 2011.

[2] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Multi-objective optimization for networks-on-chip architectures using genetic algorithms," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, 2010, pp. 3725–3728.

[3] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2015, pp. 162–163.

[4] A. Tran and B. Baas, "Noctweak: A highly parameterizable simulator for early exploration of performance and energy efficiency of networks on-chip," *Dept. Electr. Comput. Eng., Univ. California, Davis, CA, USA, Tech. Rep. ECE-VCL-2012-2*, 2012.

[5] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. Dally, "Booksim interconnection network simulator," http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim, visited 2017-02-03.

[6] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "Garnet: A detailed on-chip network model inside a full-system simulator," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 33–42.

[7] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "A topology-based design methodology for networks-on-chip applications," in *2007 2nd International Design and Test Workshop*. IEEE, 2007, pp. 61–65.

[8] H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Power consumption of 3D networks-on-chips: Modeling and optimization," *Microprocessors and Microsystems*, vol. 37, no. 6, pp. 530–543, 2013.

[9] J. Latif, S. Azam, H. N. Chaudhry, and T. Muhammad, "Performance evaluation of modern network-on-chip router architectures," *Int. J. Com. Dig. Sys*, vol. 5, no. 2, 2016.

[10] R. K. Saini and M. Ahmed, "2D hexagonal mesh Vs 3D mesh network on chip: A performance evaluation," *Int. J. Com. Dig. Sys*, vol. 4, no. 1, 2015.

[11] H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Modeling and implementation of an output-queuing router for networks-on-chips," pp. 241–248, 2007.

[12] K. Aisopos, C.-H. O. Chen, and L.-S. Peh, "Enabling system-level modeling of variation-induced faults in networks-on-chips," in *Proceedings of the 48th Design Automation Conference*. ACM, 2011, pp. 930–935.

[13] M. R. Seifi and M. Eshghi, "A clustered noc in group communication," in *TENCON 2008-2008 IEEE Region 10 Conference*. IEEE, 2008, pp. 1–5.

[14] U. Saravanakumar, R. Rangarajan, R. Haripriya, R. Nithya, and K. Rajasekar, "Cluster based hierarchical routing algorithm for network on chip," *Circuits and Systems*, vol. 4, no. 05, p. 401, 2013.

[15] F. Ge, N. Wu, X. Qin, and Y. Zhang, "Clustering-based topology generation approach for application-specific network on chip," in *Proceedings of the world congress on engineering and computer science*, vol. 2, 2011.

[16] Y. Cui, W. Zhang, and H. Yu, "Decentralized agent based re-clustering for task mapping of tera-scale network-on-chip system," in *2012 IEEE International Symposium on Circuits and Systems*. IEEE, 2012, pp. 2437–2440.

[17] G. Castilhos, M. Mandelli, G. Madalozzo, and F. Moraes, "Distributed resource management in NoC-based mpsocs with dynamic cluster sizes," in *2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2013, pp. 153–158.

[18] H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Introducing OperaNP: a reconfigurable NoC-based platform," in *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on*. IEEE, 2007, pp. 940–943.

[19] Z. Lu, L. Xia, and A. Jantsch, "Cluster-based simulated annealing for mapping cores onto 2D mesh networks on chip," in *Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on*. IEEE, 2008, pp. 1–6.

[20] A. T. Tran and B. M. Baas, "Roshaq: High-performance on-chip router with shared queues," in *Computer Design (ICCD), 2011 IEEE 29th International Conference on*. IEEE, 2011, pp. 232–238.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[22] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "Power optimization for application-specific networks-on-chips: A topology-based approach," *Microprocessors and Microsystems*, vol. 33, no. 5, pp. 343–355, 2009.

[23] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Area-aware topology generation for application-specific networks-on-chip using network partitioning," in *2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. IEEE, 2009, pp. 979–984.

[24] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebalis, "A reliability-aware design methodology for networks-on-chip applications," in *Design & Technology of Integrated Systems in Nanoscal Era, 2009. DTIS'09. 4th International Conference on*. IEEE, 2009, pp. 107–112.

[25] A. Morgan, H. Elmiligi, M. El-Kharashi, and F. Gebali, "Bio-inspired NoC architecture optimization," in *Autonomic*

Networking-on-Chip: Bio-Inspired Specification, Development, and Verification, P. Cong-Vinh, Ed. CRC Press, 2012, pp. 21–45.

[26] A. A. Morgan, H. Elmiligi, F. Gebali, and M. W. El-Kharashi, "Unified multi-objective mapping and architecture customisation of networks-on-chip," *IET Computers & Digital Techniques*, vol. 7, no. 6, pp. 282–293, 2013.

[27] Y. Z. Tei, Y. W. Hau, N. Shaikh-Husin, and M. N. Marsono, "Network partitioning domain knowledge multiobjective application mapping for large-scale network-on-chip," *Applied Computational Intelligence and Soft Computing*, vol. 2014, p. 9, 2014.

[28] P. Gorski and D. Timmermann, "Centralized traffic monitoring for online-resizable clusters in networks-on-chip," in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2013 8th International Workshop on*. IEEE, 2013, pp. 1–8.

[29] R. Manevich, I. Cidon, and A. Kolodny, "Dynamic traffic distribution among hierarchy levels in hierarchical networks-on-chip (NoCs)," in *Networks on Chip (NoCS), 2013 Seventh IEEE/ACM International Symposium on*. IEEE, 2013, pp. 1–8.

[30] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Networks-on-chip architecture customization using network partitioning: A system-level performance evaluation," *International Journal of Computing and Digital Systems*, vol. 4, no. 1, pp. 19–31, Jan 2015.

[31] A. S. Hassan, A. A. Morgan, and M. W. El-Kharashi, "An enhanced network-on-chip simulation for cluster-based routing," *Procedia Computer Science*, vol. 94, pp. 410–417, 2016.

Ahmed S. Hassan Ahmed S. Hassan received B.Sc. degree in systems and biomedical engineering, Cairo University, Egypt, in 2011. He is an embedded software developer, specialized in multicore architecture, wireless connectivity, and automotive Ethernet. Currently an M.Sc. candidate at Ain Shams University, Cairo, working on many-core Systems-on-Chip (SoC) analysis and design.

Ahmed A. Morgan Ahmed A. Morgan received the Ph.D.degree from the University of Victoria, Victoria, BC, Canada, in 2011, and the B.Sc. degree (first class honors) and the M.Sc. degree from the Faculty of Engineering at Shoubra, Benha University, Egypt in 2000 and 2005, respectively. He got a Diploma in Electronic Design Automation (EDA) and VLSI Design from the Information Technology Institute (ITI), Cairo, Egypt in 2002. He is an Assistant Professor in the Department of Computer Engineering, Cairo University, Egypt. His research interests include parallel architectures, multicore systems, digital VLSI design, wireless sensor networks, and Networks-on-Chip (NoC) modeling, optimization, and performance evaluation.

M. Watheq El-Kharashi M. Watheq El-Kharashi received the Ph.D. degree in computer engineering from the University of Victoria, Victoria, BC, Canada, in 2002, and the B.Sc. degree (first class honors) and the M.Sc. degree in computer engineering from Ain Shams University, Cairo, Egypt, in 1992 and 1996, respectively. He is a Professor in the Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt and an Adjunct Professor in the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. His general research interests are in advanced system architectures, especially Networks-on-Chip (NoC), Systems-on-Chip (SoC), and secure hardware. He published about 100 papers in refereed international journals and conferences and authored two books and 6 book chapters.