Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2010, Article ID 603059, 12 pages
doi:10.1155/2010/603059

Research Article

3D Network-on-Chip Architectures Using Homogeneous Meshes and Heterogeneous Floorplans

Vitor de Paulo and Cristinel Ababei

Electrical and Computer Engineering Department, North Dakota State University, Fargo, ND 58108-6050, USA

Correspondence should be addressed to Cristinel Ababei, cristinel.ababei@ndsu.edu

Received 19 February 2010; Accepted 25 August 2010

Academic Editor: Lionel Torres

Copyright © 2010 V. de Paulo and C. Ababei. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose new 3D 2-layer and 3-layer NoC architectures that utilize *homogeneous* regular mesh networks on a separate layer and one or two *heterogeneous* floorplanning layers. These architectures combine the benefits of compact heterogeneous floorplans and of regular mesh networks. To demonstrate these benefits, a design methodology that integrates floorplanning, routers assignment, and cycle-accurate NoC simulation is proposed. The implementation of the NoC on a separate layer offers an additional area that may be utilized to improve the network performance by increasing the number of virtual channels, buffers size, or mesh size. Experimental results show that increasing the number of virtual channels rather than the buffers size has a higher impact on network performance. Increasing the mesh size can significantly improve the network performance under the assumption that the clock frequency is given by the length of the physical links. In addition, the 3-layer architecture can offer significantly better network performance compared to the 2-layer architecture.

1. Introduction

3D integration is emerging as an attractive solution to the problem of increasing global interconnect delay of integrated systems [1–4]. The main advantage of 3D integration technologies is that the footprint area of the chip is smaller compared to the 2D case. Therefore, because the connections between device layers can be realized by short and reduced delay through silicon vias (TSVs), the average interconnect delay is significantly shorter. However, 3D integration technologies face challenges related to thermal issues.

Network-on-Chip (NoC) represents a new design paradigm for increasingly complex Systems-on-Chip (SoC), and since the idea of routing packets instead of wires was proposed [5–7], it has grown into a rich research topic [8–10]. The NoC concept replaces design-specific global on-chip wires with a generic on-chip interconnection network realized by specialized routers that connect generic processing elements (PE)—such as processors, ASICs, FPGAs, memories, and so forth—to the network and facilitate communications or links between them. The benefits of the NoC based SoC-design include scalability, predictability, and higher bandwidth with support for concurrent communications.

The scalability and predictability of NoCs enable designers to design increasingly complex systems, with large numbers of IP/cores and lower communication latencies for many applications. In such scenarios, where flexibility and predictability are primary concerns, homogeneous regular networks (see Figure 1(a)) are preferred. However, homogeneous NoC topologies have limitations in that communication locality is poorly supported, and the utilization of network resources is low. Moreover, designs with IP/cores with different sizes are not well suited to implementations based on regular mesh NoC topologies. Therefore, when area and performance are more important, application-specific heterogeneous irregular networks (see Figure 1(b)) are preferred. However, the design of these networks is more difficult and specialized routing algorithms are necessary to prevent deadlock.

Most of the previous works assumed equal area for all tiles. This assumption simplifies the design process due to the regularity of the mesh NoC topologies. However, assuming tiles with equal area in cases where IP/cores have different

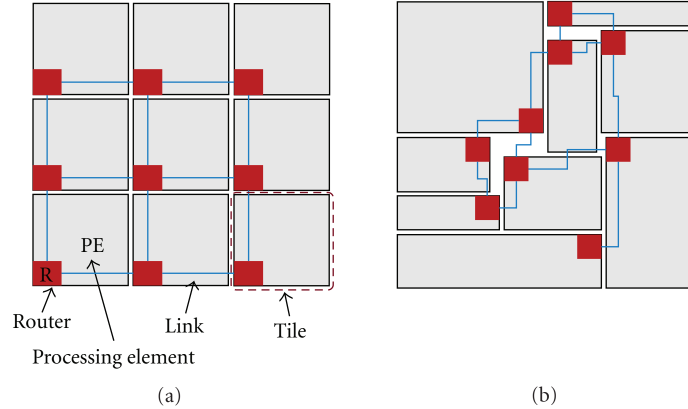


Figure 1: (a) 2D homogeneous NoC design. A tile is composed of a router (R) and a generic processing element (PE). A PE can implement any IP/core of a given application. Routers are interconnected via physical links. (b) Custom heterogeneous NoC design.

sizes is unrealistic. To address this problem for systems with heterogeneous floorplans and to exploit the benefits of 3D integration, in this paper, we propose new 3D NoC architectures. The proposed architectures utilize homogeneous networks on a separate layer and heterogeneous floorplans on different layers. Our objective is to combine the benefits of compact heterogeneous floorplans with those of regular homogeneous mesh networks.

2. Related Work

In this section, we discuss recent works on 3D NoCs and studies that utilize floorplanning information during the design and optimization of NoCs. The reader is referred to [11, 12] for recent surveys of NoCs. Nanophotonic and wireless NoCs have also been recently proposed as alternative solutions to 2D architectures. However, they pose several issues including scaling and integration of photonic devices and power dissipation of mm-wave transceivers [13].

Performance benefits of 3D NoC topologies were investigated analytically in [14] and experimentally in [15]. The transition from 2D to 3D NoC architectures is done by equally distributing tiles onto the device layers of the 3D architecture in [16, 17]. A different approach is to reduce the footprint of each tile by implementing the processing element and the router in a distributed fashion across layers [18]. By leveraging long wires to connect remote intralayer nodes, a low-diameter 3D network is studied in [19]. Efficient application-specific 3D NoC topology synthesis algorithms are studied in [20]. A novel layer-multiplexed 3D network architecture with vertical demultiplexing and multiplexing links is proposed in [21]. The scalability of 3D NoCs in terms of throughput, latency, and area overhead is studied in [22]. The per-flow worst-case communication performance in 2D and 3D regular mesh NoCs with four layers is investigated in [23]. The first demonstration of a fabricated 3D NoC is reported in [24]. Previous works focused on homogeneous regular NoCs and floorplans where all PEs have equal size. In this paper, the proposed architectures provide a design solution to applications where PEs are heterogeneous with different sizes. Hence, the proposed 3D NoC architectures represent an alternative solution to heterogeneous irregular NoC topologies.

Floorplanning information is used in the area-wirelength calculations from [25, 26] or during mapping [27]. A floorplanner is used to compute links power consumption and to detect timing violations in application-specific NoCs in [28]. The NoC synthesis approach from [28] was extended to designing custom 3D network topologies in [29]. Slicing floorplanning is used in the design methods for custom NoC topology synthesis studied in [30–32]. Frequently communicating modules are placed next to each other using a floorplanner in the network synthesis heuristic studied in [33]. A physical planner is used in [34] during topology design to reduce power consumption on wires. Previous works use floorplanning either for wirelength and power estimations or to find a *single* placement, which then remains fixed throughout the topology synthesis process. However, other floorplanning solutions may represent better starting placements for the synthesis process. To address this problem, the methodology proposed in this paper explores multiple floorplans to increase the chance of finding the optimal initial placement.

3. Contribution

In this paper, we propose novel 3D NoC architectures and implement an automated design space-exploration tool. Our main contribution can be summarized as follows.

(i) We propose and study two 3D NoC architectures (2-layer and 3-layer architectures) that utilize a homogeneous network on a separate layer and heterogeneous floorplans on different layers. In this way, the network regularity is maintained for flexibility and delay predictability while the IP/cores can have arbitrary sizes. This approach avoids design difficulties due to IP/core size irregularities that are typically addressed by specialized routing algorithms [35].

(ii) For the 2-layer architecture, we propose the use of a floorplanning and routers assignment-based design methodology for the placement of IP/cores on the

first layer and the minimization of their connections to the NoC routers located on the second layer. In the case of the 3-layer architecture, the design methodology also includes a partitioning step. The floorplanning of the two partitions on layers 1 and 3 is done using a newly modified floorplanner capable of handling vertical constraints. One advantage of the separation between IP/cores and network is that once the best floorplan is found, one can focus on improving the system performance by focusing on the network. The second layer has an additional available area that may be utilized to increase the number of routers or their complexity (e.g., increase the number of virtual channels and/or the buffers size). In addition, network interfaces (NIs), which are important components of NoC-based systems, also may be placed on the second layer, thereby reducing the footprint area of the floorplanning layers.

(iii) We implemented a versatile software framework to investigate the benefits of the proposed 3D architectures. It integrates an efficient B*Tree-based floorplanner with a cycle-accurate NoC simulator for maximum confidence in the experimental results.

Preliminary results on the 2-layer NoC architecture were reported in [36]. In this paper, we also propose the second 3-layer NoC architecture that aims at further reducing the footprint area of the chip and at improving the average flit latency. Due to the differences from the first proposed architecture, we also modify the design methodology by introducing an additional partitioning step and enhancing the floorplanning algorithm to handle vertical constraints.

4. 3D Architectures

4.1. 2-Layer Architecture. The *2-layer architecture* has two device layers. The first layer is used entirely for the *heterogeneous irregular* IP/cores, while the second layer is dedicated to the *homogeneous regular* NoC (see Figure 2(b)). This approach simplifies the design process in that it separates the floorplanning optimization from the network topology synthesis. The goal of the floorplanning step is to find the best floorplan with minimal white space. The second device layer accommodates the regular mesh network. In this way, the network regularity is maintained for flexibility and delay predictability, while the IP/cores can have arbitrary sizes. In addition, a simple packet routing algorithm can be used such as the deterministic XY routing.

4.2. 3-Layer Architecture. The *3-layer architecture* has three device layers. The second layer is again dedicated to implementing the NoC, while layers 1 and 3 are used for IP/cores placement (see Figure 2(c)). This architecture aims at reducing the footprint area of the chip, which in turn leads to shorter physical links, hence improving the network performance (overall average flit latency). In both proposed architectures, the vertical connections between IP/cores and their assigned routers are realized using through silicon vias

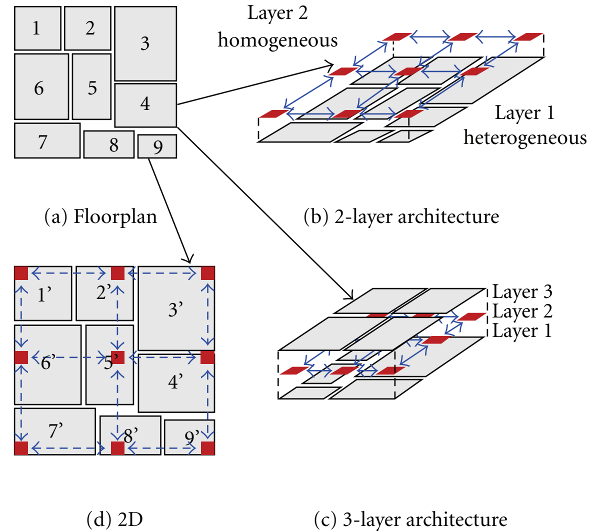


Figure 2: (a) Initial floorplan with no routers. (b) 2-layer architecture. (c) 3-layer architecture. (d) 2D implementation with each IP/core inflated to account for area required by network interfaces, routers and wires for physical links.

(TSV). Routers connected to IP/cores have five ports, while the rest of the routers have only four ports.

One advantage of the proposed 3D NoC architectures is that the 3D fabrication will be simpler compared to 3D architectures with more than three layers [17], as the misalignment is only between two or three layers. In addition, the thermal management of fewer layers also will be simpler. Moreover, the extra space available on the second layer may be used to increase the number of routers or their complexity. The additional area may be utilized to implement fault/error tolerance techniques such as error-correcting codes to address crosstalk issues or could be allocated to additional wires to increase the bandwidth of physical links and therefore improve the overall network performance. Alternatively, the extra area also may be utilized to implement thermal monitoring and management schemes [37], to implement buffers for pipelining the physical links, or to incorporate reconfigurability capabilities [38].

5. Proposed Design Methodology

The proposed design methodology is presented in Figure 3. The input to the design flow is the application represented as a communication task graph (CTG) whose tasks have been mapped to *floating* IP/cores using existing mapping algorithms [39]. By floating it is meant that the location of these tasks is yet to be determined during the floorplanning step. In addition, the user can specify several control parameters including the number of different floorplans to be explored N, the number of best floorplans M recorded in the *best M list* and evaluated later using the integrated cycle-accurate simulator and the mesh size $R \times R$. The main steps of the proposed design methodology are described in the following sections.

5.1. Partitioning of the CTG. This step is done only for the 3-layer architecture. Because in this case, the IP/cores are placed on two layers (1 and 3), we first partition the CTG of each application into two subgraphs (see Figure 4), which will be placed by the floorplanner in the next step. The bipartitioning is done such that the total area of the cores in each partition is balanced while the number of arcs (an arc represents a source-destination communication pair in the communication task graph) cut is minimized. The two partitions have to be balanced to minimize the footprint area of the 3D chip, which will be determined by the maximum area of the accumulated area of the blocks in each partition. The minimization of the cut size between the two partitions has as a result that highly connected cores are placed on the same layer. This, as observed in our experiments, helps these cores to be floorplanned closer to each other, which in turn leads to better overall latencies.

For this step, we use the well-known hMetis partitioner [40, 41]. hMetis is a multilevel move-based partitioner, which can achieve balanced and minimum cutsize partitions very efficiently.

5.2. Exploration of N and Recording of Best M Floorplans. The integrated floorplanner is based on the B*Tree representation from [42]. It employs a simulated annealing-based algorithm, with a cost function that combines area and wirelength (user can specify $\alpha \in [0, 1]$)

$$\text{Cost function} = \alpha \cdot \text{Area} + (1 - \alpha) \cdot \text{Wire Length}. \quad (1)$$

Connections between cores are weighted by the communication volume (available from the CTG) so that the resulting floorplanning solution minimizes first the connections with higher communication volume.

A number of N different floorplans are generated by running the floorplanner N times with the selected weights for area and wirelength and with different seeds for the internal random number generator. During this step, a number of best $M \leq N$ floorplans are recorded in the *best M list*. The selection is made according to the chosen criterion of smaller area or shorter total wirelength, which is related to the total communication volume inside the application. The default values of N and M are $N = 30$, $M = 10$. However, they also may be specified by the user. A typical result after floorplanning is shown in Figure 5.

For the 3-layer architecture, the floorplanning step is different. In this case, cores are placed on layers 1 and 3 as dictated by the result of the partitioning step. To do that, we have modified the floorplanning algorithm to be able to handle *vertical constraints*. As a result, this step is split into two substeps (i) In the first substep, the first partition is floorplanned on layer 1 using the original version of the floorplanning algorithm (this is similar to the 2-layer architecture). (ii) During the second sub-step, the second partition is floorplanned on layer 3 using the modified floorplanner. In this case, connections between cores of this partition and cores of the first partition (already placed and fixed on layer 1) act as vertical constraints for the floorplanning process on layer 3. Vertical constraints

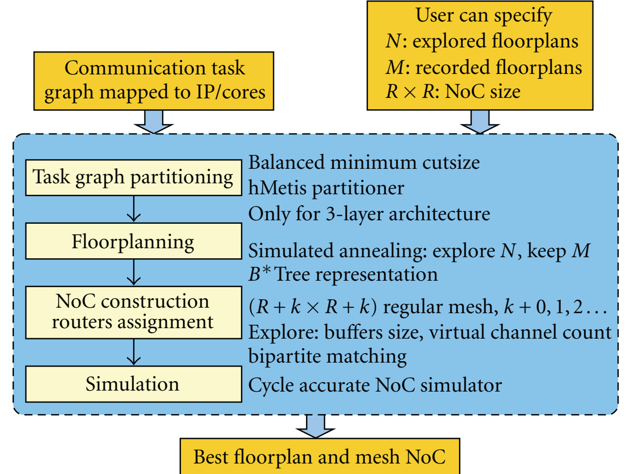
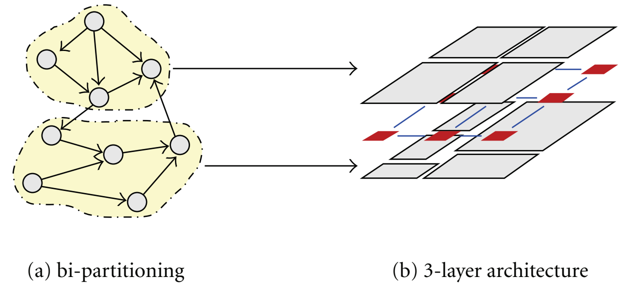


FIGURE 3: Proposed NoC design exploration methodology.



(a) bi-partitioning (b) 3-layer architecture

FIGURE 4: Bipartitioning of the application communication task graph and assignment of the two partitions to layers 1 and 3 of the 3-layer architecture.

aim at minimizing the overall wirelength of the top-level application. Intuitively, cores on layer 3 connected in the top-level communication task graph to cores on layer 1 should be overlapping or vertically aligned to shorten the communication distance via the network on layer 2. A typical result after floorplanning for the 3-layer architecture is shown in Figure 6.

5.3. Routers Assignment. In this step, each floorplan from the list of best M floorplans undergoes the routers assignment step. The regular $R \times R$ mesh NoC is constructed on layer 2. This square regular mesh network utilizes the minimum number of routers that can guarantee at least one router for each IP/core. This topology is referred to as the *direct topology*. However, the mesh can optionally be expanded to a larger number of routers in both x, y directions. Since we deal with heterogeneous floorplans, it is not possible to guarantee the presence of routers at the locations of IP/core corners (or even the IP/core layout). Therefore, some of the IP/cores will have to use *extralinks* to connect to the assigned routers. These extra-links introduce additional delays (included inside the cycle-accurate simulator) that affect the overall performance.

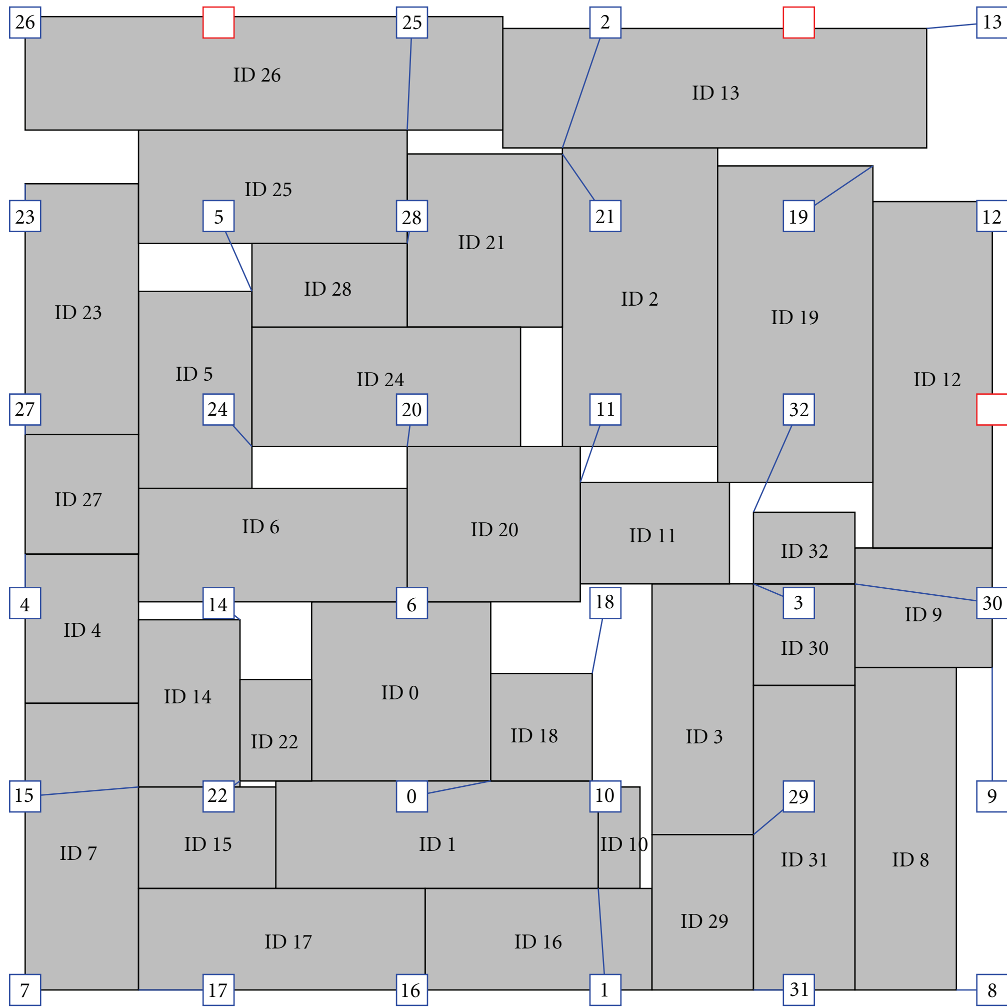


FIGURE 5: Screen capture with the result of floorplanning and routers assignment for *ami33*. The NoC is a direct topology of 6×6, which is the minimum to guarantee at least a router for each IP/core. The extralinks are shown as straight lines between cores and assigned routers.

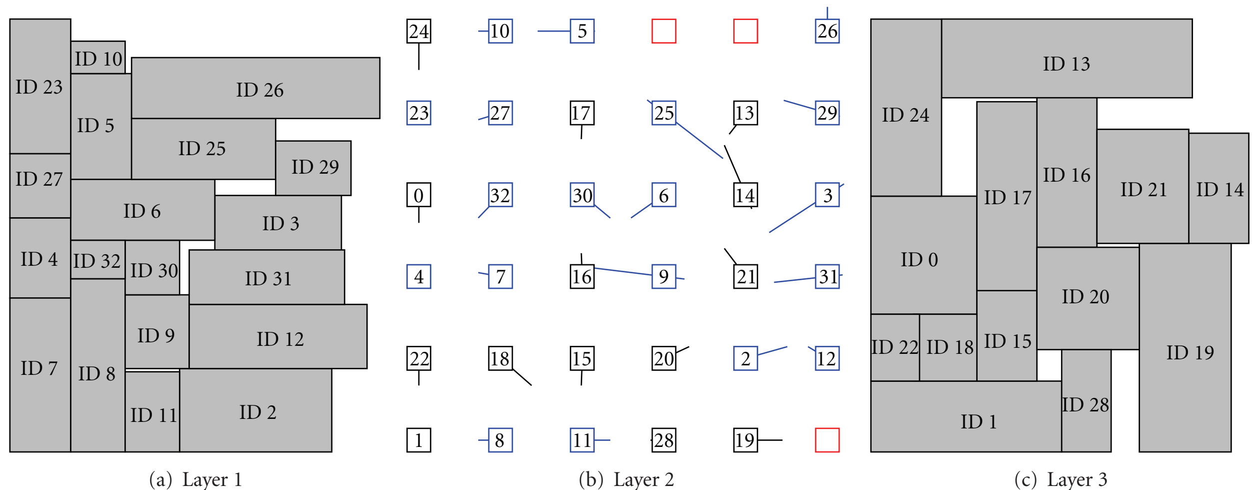


FIGURE 6: Floorplanning and routers assignment for *ami33* using the 3-layer architecture.

Table 1: Characteristics of the used testcases.

Testcase	Number of IP/cores	Core avg. W/H	Core std. dev. of W/H	Direct topology $R \times R$
apte	8	4324/2499	27/4	3×3
xerox	10	2114/2872	335/1290	4×4
hp	11	4533/924	2498/386	4×4
ami25	25	1770/1408	1201/896	5×5
ami33	33	1581/1573	830/865	6×6
ami49	49	1089/1123	768/651	7×7

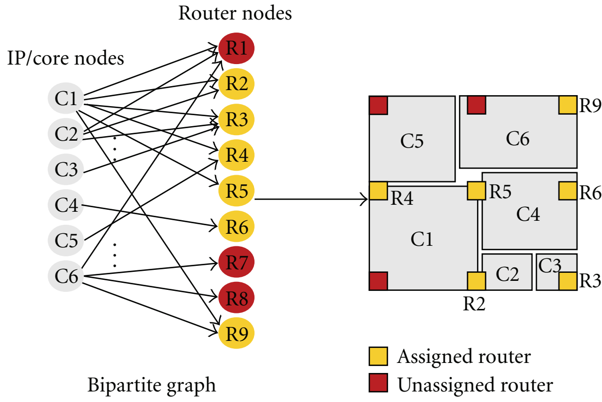


Figure 7: Illustration of the routers assignment step on a testcase with 6 IP/cores and a regular mesh network of 3×3 routers.

The goal of the routers assignment step is to associate each IP/core with a router from the regular mesh on layer 2 such that the total wirelength of the extra-links between each IP/core and its assigned router is minimized. This is a linear assignment problem solved by using the efficient Kuhn-Munkres algorithm [43]. The algorithm utilizes a bipartite graph (see Figure 7) with two sets of nodes: left-nodes representing the application IP/cores and right-nodes representing the routers of the regular mesh NoC. Edges connect each node from one set to all nodes in the other set. Edge weights are proportional to the Manhattan distance between the IP/core and routers. In this way, we treat the assignment of all IP/cores *simultaneously* and achieve an overall minimal total length of the extra-links. This step is the same for both 2-layer and 3-layer architectures. The examples from Figures 5 and 6 also show the result of the routers assignment step.

5.4. NoC Simulation. In the last step, each of the best M NoC topologies is verified using the integrated cycle-accurate simulator. The simulator is an adapted version of the one studied in [37]. We use the following default values for the NoC topology: packet size of 5 flits with each flit being 64 bits wide, input buffer size of 12 flits, and two virtual channels. We use XY routing and wormhole flow control, which is known to be very efficient and requiring small hardware overheads. The cycle-accurate simulator is always run until all injected flits reached their destination and the average latency is computed allowing first 1000 warm-up cycles. The router architecture is similar to the one presented in [44]. The final average flit latency, which is obtained during this step, is recorded for each of the floorplans from the best M list. The NoC topology with the best overall latency is selected as the final result.

Finally, we note that ideally, one would use the routers assignment and the cycle-accurate simulation inside the optimization loop of the simulated annealing based floorplanning algorithm (the concept of unifying different design flow steps to better explore the design solution space has been applied successfully for example to mapping and routing in [45].) However, this becomes computationally too expensive due to the long CPU runtimes required by the cycle-accurate simulator.

6. Experimental Results

6.1. Experimental Setup and Testcases. We implemented the proposed design methodology, which integrates the partitioner, the floorplanner, the routers assignment, the NoC cycle-accurate simulator, and the GUI, using C++. The tool can be downloaded from [46]. In our experiments, we used six testcases whose characteristics are shown in Table 1. In this table, we also present the size of the *direct topologies*. We constructed these testcases from the classic MCNC testcases, whose area was scaled to achieve an average size of about 1 cm $\times$ 1 cm, which is a typical area for NoCs reported in the literature [24, 47–49]. The initial connectivity between the modules was used to compute the communication volume in the communication task graph associated with each testcase floorplan.

For the simulated annealing-based floorplanning step, we used an *alpha* value of 0.25, which in our experiments proved to be a good balance between area and wirelength while the aspect ratio of the resulting floorplan was close to 1. In the NoC simulation step, each testcase was subject to uniform traffic with packets injected at each source router at a rate proportional to the communication volume of the corresponding source-destination communication pair.

Because in our methodology the length of the physical links between the network routers varies with the network size, we estimate the link delay by extrapolating the physical link delay from [37] using a simple Elmore delay formula [50]. The same delay estimation technique was applied to the extra-links between IP/cores and routers, which were

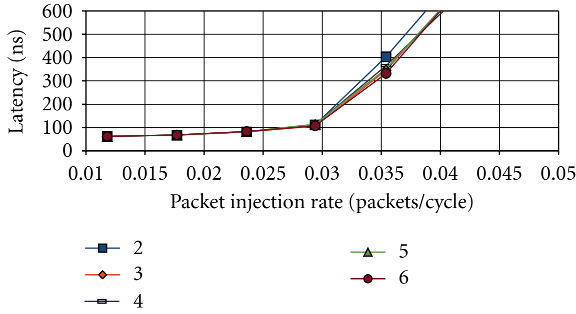


Figure 8: Latency as number of virtual channels is varied for *apte*.

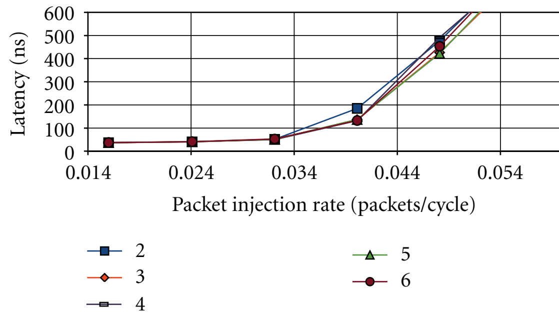


Figure 9: Latency as number of virtual channels is varied for *xerox*.

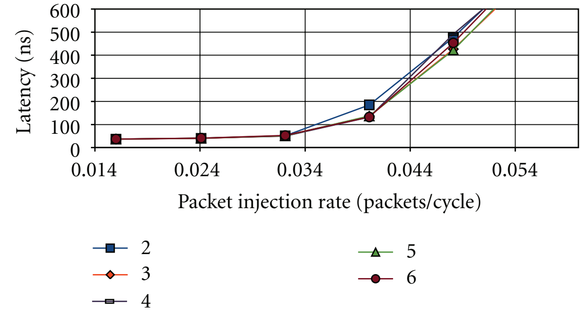


Figure 10: Latency as number of virtual channels is varied for *hp*.

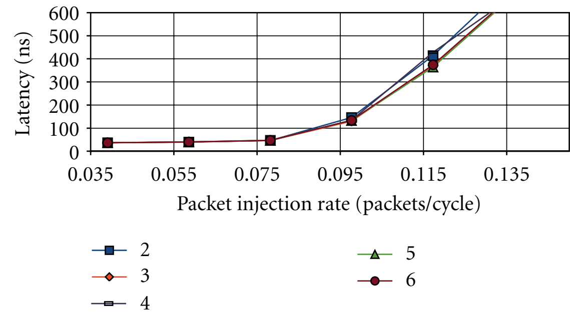


Figure 11: Latency as number of virtual channels is varied for *ami25*.

assumed to be L-shaped (with negligible via delay between metal layers). We do, however, consider the delay of the through silicon vias (TSVs) between two device layers of the 3D architectures. We estimated the TSV delay by technology projection [51] using the delay data from [17]. Based on the analyses in [17, 52], we assume that the area required by TSVs is negligible and that TSVs can be accommodated within the white space available in typical floorplans. The CPU runtime is approximately 30 minutes (Linux machine, 2.5 GHz, 2 GB memory) for the largest testcase *ami49*.

6.2. Exploration of the 2-Layer Architecture. In the first part of the experiments, several variations are applied to the default network specifications. The purpose of these experiments is to identify the optimal network that minimizes the average flit latency.

6.2.1. Varying the Virtual Channels Count. We start by investigating the impact of increasing the number of virtual channels. We can afford to do that because routers are expected to be smaller than the average core size (roughly 20% of the total cores area), which means that on layer 2 there is extra area that may be utilized to further improve the NoC performance. In this experiment, we varied the number of virtual channels between 2 and 6.

The results for the average flit latency (as reported by the cycle-accurate NoC simulator) are shown in Figures 8, 9, 10, 11, 12, and 13. We observe that the average flit latency generally improves with the increase of the number of virtual channels. We also plot (see Figure 14) the normalized latency

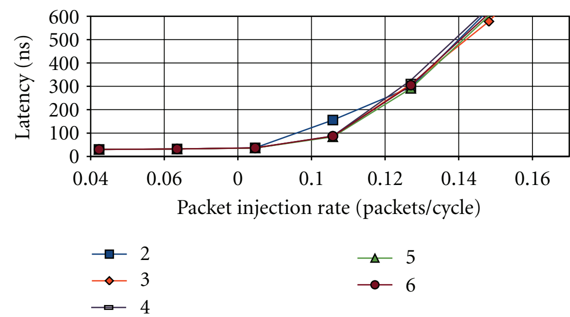


Figure 12: Latency as number of virtual channels is varied for *ami33*.

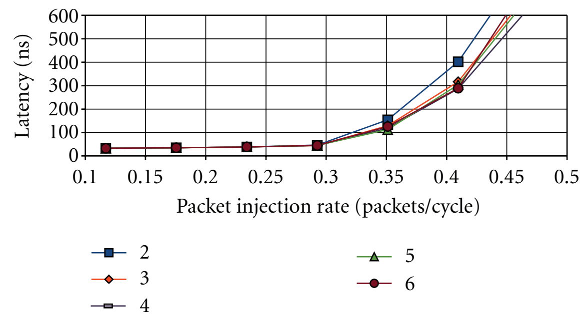


Figure 13: Latency as number of virtual channels is varied for *ami49*.

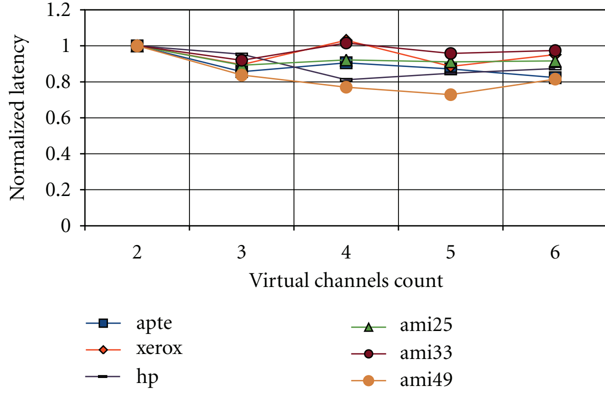


Figure 14: Normalized latency with variation of the number of virtual channels for the packet injection rate when the network saturation occurs.

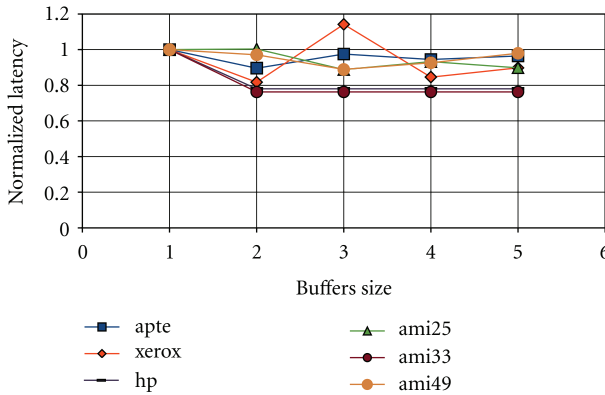


Figure 15: Normalized latency with variation of the buffers size for the packet injection rate when the network saturation occurs.

(with respect to the latency achieved using the default of 2 virtual channels) for the packet injection rate when the network saturation occurs. We find that the optimal number of virtual channels is different for different testcases.

These results are expected because the overall congestion in the network is intuitively reduced if the number of virtual channels multiplexed in the time-domain over a physical channel is increased. However, it is also evident that increasing the number of virtual channels more than necessary can have a negative impact on performance. This is explained as follows: as the network gets loaded with injected packets, the average amount of stalling due to arbitration inside routers (with increasingly more virtual channels) may increase and affect negatively the latency.

6.2.2. Varying the Buffers Size. In this experiment, instead of increasing the number of virtual channels, we increase the buffers size. Since we assume the area occupied by routers to be roughly 20% of the total cores area, we increase the area of each router to up to 5x by increasing both the input and output buffer sizes of each port (buffers occupy most of the area inside the router architecture).

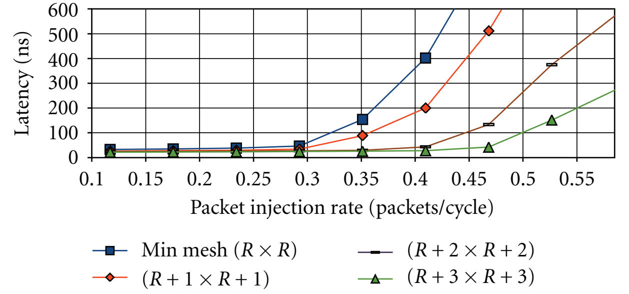


Figure 16: Latency as mesh size is varied for *ami49*.

Due to space limitations, we report (see Figure 15) only the normalized latency (with respect to the latency achieved using the minimum buffer size) for the packet injection rate when the network saturation occurs. We observe that the average flit latency improves with the increase of the buffer size. However, once a certain buffer size is reached (which is roughly the 3x data point except for *xerox*) further increasing the buffer size does not improve the latency. We note that in general, latency is improved more by increasing the number of virtual channels rather than increasing the buffers size.

6.2.3. Varying the Mesh Size. In this experiment, we investigate the impact of increasing the size of the regular mesh network ($R + k \times R + k$, $k = 1, 2, 3$) on the average flit latency. Increasing the number of routers in both dimensions x, y for the same testcase translates in shorter physical links between routers. As a result, the entire system can be clocked at higher frequencies, which significantly improves the saturation throughput. For example, the result of this experiment for testcase *ami49* is shown in Figure 16. The results of the rest of the testcases are similar; we do not include the rest of the plots here due to space limitations.

It has to be noted that this result is achieved under the assumption that the delay of the router pipeline is a clock period, which is given by the delay of the physical link. This assumption is made in [37], from where we adopted the NoC simulator, and can be validated by using speculation and lookahead as discussed in [8]. If, however, this assumption is removed, the router pipeline should incur a delay equal to two, three, or four clock cycles (to account for typical operations including routing computation, virtual channel allocation, switch allocation, and switch traversal), depending on the type of flit (head, body or tail) and the degree of speculation and lookahead [8]. On the other hand, if the router pipeline is assumed to incur a constant and fixed delay—irrespective of the physical link length—then the system clock frequency will be given by the maximum between the delay of the physical link and the delay of the router pipeline. In this case, the impact of increasing the mesh size will be less significant, and it could actually lead to an increase of the flit latency as studied in [36].

6.3. Comparison of the 2-Layer and 2D Architectures. In order to compare the proposed 2-layer architecture against

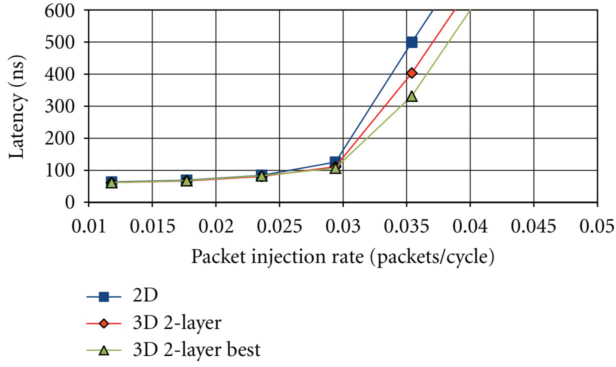


FIGURE 17: Latency comparison for 2D and 2-layer implementations of *apte*.

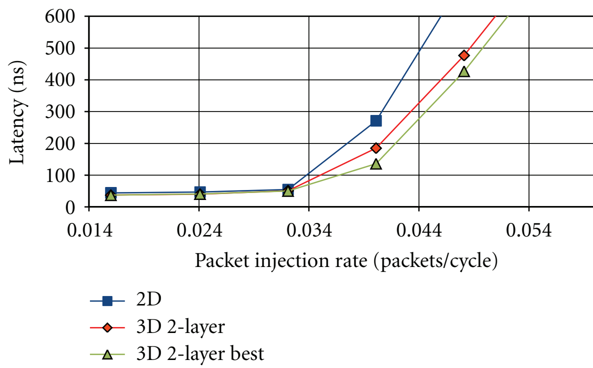


FIGURE 18: Latency comparison for 2D and 2-layer implementations of *xerox*.

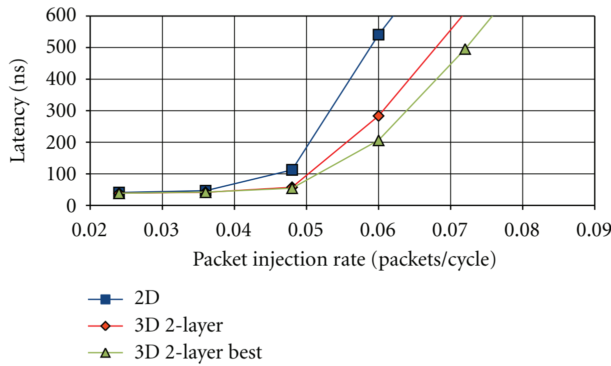


FIGURE 19: Latency comparison for 2D and 2-layer implementations of *hp*.

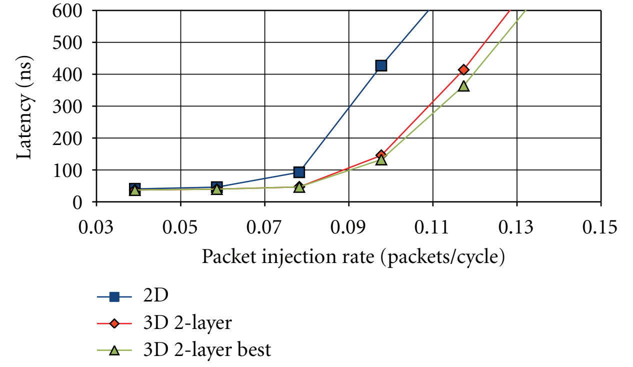


FIGURE 20: Latency comparison for 2D and 2-layer implementations of *ami25*.

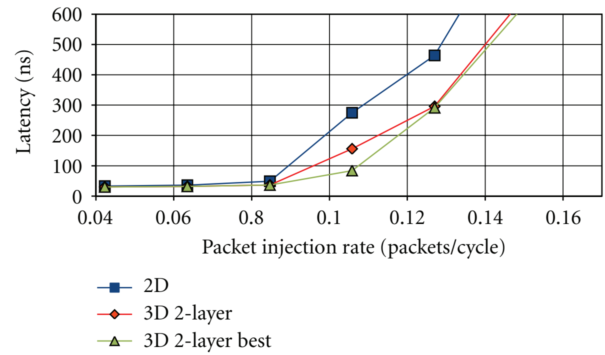


FIGURE 21: Latency comparison for 2D and 2-layer implementations of *ami33*.

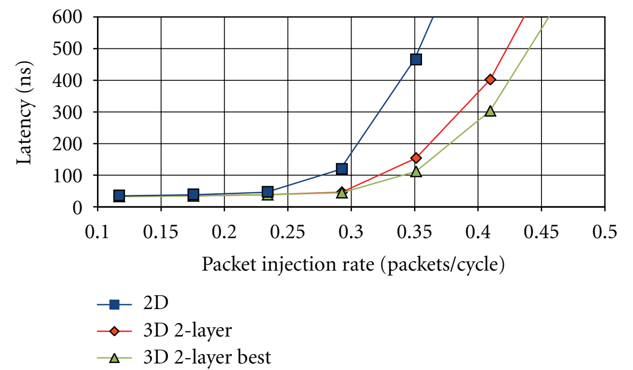


FIGURE 22: Latency comparison for 2D and 2-layer implementations of *ami49*.

a traditional approach we construct the 2D architecture by artificially expanding the IP/cores with 20% [12, 16] to account for the space occupied by routers and network interface (implemented within the cores boundaries on the same layer).

The simulation results are shown in Figures 17, 18, 19, 20, 21, and 22. We observe that the performance of the 2-layer architecture is better for each testcase. For the 2-layer architecture, the additional delays incurred due to the TSVs negatively impact the flit latency. However, the footprint area is smaller (as cores are smaller), and therefore the physical links are shorter, which leads to significantly smaller flit latencies. In addition, if we exploit the extra area on the top layer of the 2-layer architecture as described in the previous sections, then the latency can be further improved as illustrated by the *3D 2-layer best* data points from Figures 17–22.

We also note that the performance of the 2D architecture may be improved by designing custom NoCs similar to those studied in [28, 30]. However, the main goal of this paper is not to show that *regular* homogeneous 3D NoCs are better than *custom* 2D NoCs, which is unlikely, but to propose 3D

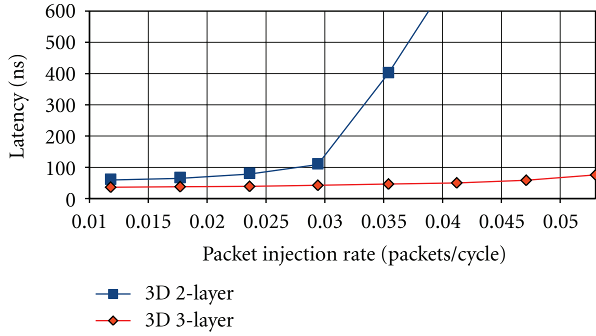


Figure 23: Latency for 2-layer and 3-layer implementations of *apte*.

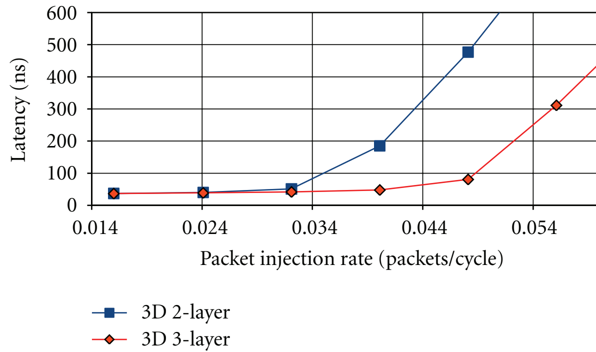


Figure 24: Latency for 2-layer and 3-layer implementations of *xerox*.

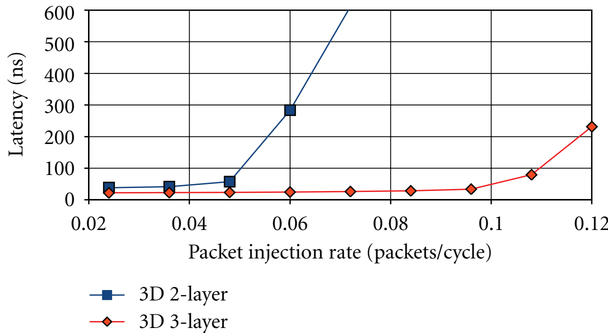


Figure 25: Latency for 2-layer and 3-layer implementations of *hp*.

architectures as alternatives to *regular* 2D NoCs and explore their performance when the number of virtual channels, buffers size, and mesh size is varied to use the extra space available on layer 2.

6.4. Comparison of the 2-Layer and 3-Layer Architectures. In the last part of our experiments, we compare the average flit latencies of the 2-layer and 3-layer architectures. The simulation results, using the default values for mesh size, number of virtual channels and buffers size are shown in Figures 23, 24, 25, 26, 27, and 28. As expected, because in the case of the 3-layer architecture the physical links are shorter, the clock frequency at which the system can work is higher. Hence, the average flit latency is improved significantly. Therefore, we conclude that the 3-layer architecture is better than the

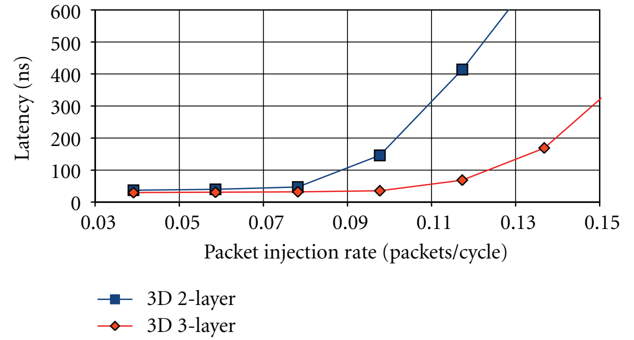


Figure 26: Latency for 2-layer and 3-layer implementations of *ami25*.

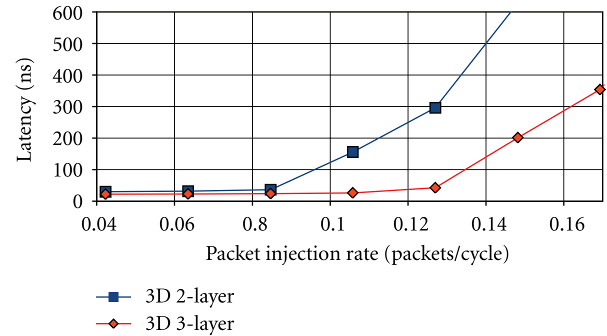


Figure 27: Latency for 2-layer and 3-layer implementations of *ami33*.

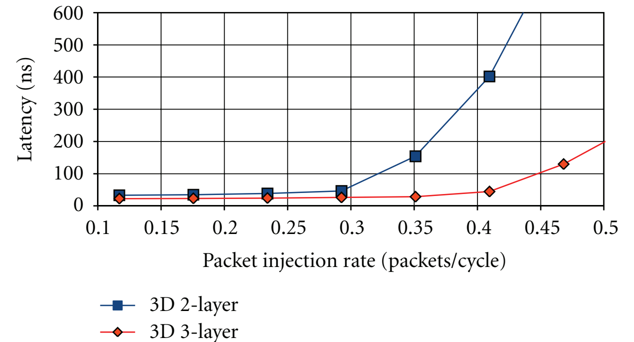


Figure 28: Latency for 2-layer and 3-layer implementations of *ami49*.

2-layer architecture. However, the design methodology for the 3-layer architecture requires additionally the partitioning step and the modification of the floorplanning algorithm. It may potentially require more complex thermal management too due to the increased integration density and the 3D fabrication technology will be more complex due to the alignment of three layers.

7. Conclusion and Future Work

In this paper, we proposed 3D 2-layer and 3-layer NoC architectures that utilize homogeneous networks on a separate layer and heterogeneous floorplans on different layers. A

design methodology that consists of floorplanning, routers assignment, and cycle-accurate NoC simulation was implemented and utilized to investigate the new architectures. Experimental results showed that increasing the number of virtual channels rather than the buffers size is more effective in improving the NoC performance. In addition, increasing the mesh size can significantly improve the NoC performance under the assumption that the clock frequency is given by the length of the physical links. Moreover, the 3-layer architecture can offer significantly better NoC performance compared to the 2-layer architecture.

As future work, we plan to address the problems of energy consumption and thermal profile optimization [53, 54] possibly in a unified fashion inside the floorplanning algorithm. The floorplanning step will be modified to consider the allocation of white space and TSVs planning under area constraints.

Acknowledgment

This paper was supported by the Electrical and Computer Engineering Department at North Dakota State University (NDSU).

References

[1] L. Xue, C. C. Liu, H.-S. Kim, S. K. Kim, and S. Tiwari, "Three-dimensional integration: technology, use, and issues for mixed-signal applications," *IEEE Transactions on Electron Devices*, vol. 50, no. 3, pp. 601–609, 2003.

[2] W. R. Davis, J. Wilson, S. Mick et al., "Demystifying 3D ICs: the pros and cons of going vertical," *IEEE Design and Test of Computers*, vol. 22, no. 6, pp. 498–510, 2005.

[3] P. Morrow, B. Black, M. J. Kobrinsky et al., "Design and fabrication of 3D microprocessors," in *Proceedings of Materials Research Society Symposium*, 2006.

[4] S. J. Koester, A. M. Young, R. R. Yu et al., "Wafer-level 3D integration technology," *IBM Journal of Research and Development*, vol. 52, no. 6, pp. 583–597, 2008.

[5] P. Guerrier and A. Grenier, "A generic architecture for on-chip packetswitched interconnections," in *Proceedings of ACM/IEEE Design Automation and Test in Europe Conference (DATE '00)*, pp. 250–256, 2000.

[6] A. Hemani, A. Jantsch, S. Kumar et al., "Network on chip: an architecture for billion transistor era," in *Proceedings of IEEE NorChip Conference*, November 2000.

[7] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, June 2001.

[8] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2004.

[9] G. de Micheli and L. Benini, *Networks on Chip*, Morgan Kaufmann, 2006.

[10] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, 2009.

[11] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, pp. 71–121, 2006.

[12] E. Salminen, A. Kulmala, and T. D. Hamalainen, "Survey of Network-on-Chip proposals," White Paper OCP-IP, 2008.

[13] L. P. Carloni, P. Pande, and Y. Xie, "Networks-on-chip in emerging interconnect paradigms: advantages and challenges," in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NoCS '09)*, pp. 93–102, May 2009.

[14] V. F. Pavlidis and E. G. Friedman, "3-D topologies for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1081–1090, 2007.

[15] B. S. Feero and P. P. Pande, "Networks-on-chip in a three-dimensional environment: a performance evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32–45, 2009.

[16] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir, "Design and management of 3D chip multiprocessors using network-in-memory," in *Proceedings of the 33rd International Symposium on Computer Architecture (ISCA '06)*, pp. 130–141, June 2006.

[17] J. Kim, C. Nicopoulos, D. Park et al., "A novel dimensionally-decomposed router for on-chip communication in 3D architectures," in *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA '07)*, pp. 138–149, June 2007.

[18] D. Park, S. Eachempati, R. Das et al., "MIRA: a multi-layered on-chip interconnect router architecture," in *Proceedings of the 35th International Symposium on Computer Architecture (ISCA '08)*, pp. 251–261, June 2008.

[19] Y. Xu, Y. Du, B. Zhao, X. Zhou, Y. Zhang, and J. Yang, "A low-radix and low-diameter 3D interconnection network design," in *Proceedings of the 15th IEEE International Symposium on High Performance Computer Architecture (HPCA '09)*, pp. 30–42, Raleigh, NC, USA, February 2009.

[20] S. Yan and B. Lin, "Design of application-specific 3D networks-on-chip architectures," in *Proceedings of the 26th IEEE International Conference on Computer Design (ICCD '08)*, pp. 142–149, October 2008.

[21] R. S. Ramanujam and B. Lin, "A layer-multiplexed 3D on-chip network architecture," *IEEE Embedded Systems Letters*, vol. 1, no. 2, pp. 50–55, 2009.

[22] A. Y. Weldezion, M. Grange, D. Pamunuwa et al., "Scalability of network-on-chip communication architecture for 3-D meshes," in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NoCS '09)*, pp. 114–123, May 2009.

[23] Y. Qian, Z. Lu, and W. Dou, "From 2D to 3D NoCs: a case study on worst-case communication performance," in *Proceedings of ACM/IEEE International Conference on Computer Aided Design (ICCAD '09)*, pp. 555–562, November 2009.

[24] C. Mineo, R. Jenkal, S. Melamed, and W. Rhett Davis, "Inter-die signaling in three dimensional integrated circuits," in *Proceedings of IEEE Custom Integrated Circuits Conference (CICC '08)*, pp. 655–658, September 2008.

[25] S. Murali and G. De Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 914–919, June 2004.

[26] J. Xu, W. Wolf, J. Henkel, and S. Chakradhar, "A design methodology for application-specific networks-on-chip," *Transactions on Embedded Computing Systems*, vol. 5, no. 2, pp. 263–280, 2006.

[27] S. Murali, L. Benini, and G. de Micheli, “Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees,” in *Proceedings of ACM/IEEE Asia and South Pacific Design Automation Conference (ASPDAC ’05)*, pp. 27–32, 2005.

[28] S. Murali, P. Meloni, F. Angiolini et al., “Designing application-specific networks on chips with floorplan information,” in *Proceedings of International Conference on Computer-Aided Design (ICCAD ’06)*, pp. 355–362, November 2006.

[29] S. Murali, C. Seiculescu, L. Benini, and G. De Micheli, “Synthesis of networks on chips for 3D systems on chips,” in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC ’09)*, pp. 242–247, Yokohama, Japan, January 2009.

[30] K. Srinivasan and K. S. Chatha, “A low complexity heuristic for design of custom network-on-chip architectures,” in *Proceedings of Design, Automation and Test in Europe (DATE ’06)*, pp. 130–135, March 2006.

[31] K. Srinivasan, K. S. Chatha, and G. Konjevod, “Linear-programming-based techniques for synthesis of network-on-chip architectures,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 4, pp. 407–420, 2006.

[32] K. S. Chatha, K. Srinivasan, and G. Konjevod, “Automated techniques for synthesis of application-specific network-on-chip architectures,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1425–1438, 2008.

[33] C. Neeb and N. Wehn, “Designing efficient irregular networks for heterogeneous systems-on-chip,” *Journal of Systems Architecture*, vol. 54, no. 3-4, pp. 384–396, 2008.

[34] T. Ahonen, D. A. Sigüenza-Tortosa, H. Bin, and J. Nurmi, “Topology optimization for application-specific networks-on-chip,” in *Proceedings of International Workshop on System Level Interconnect Prediction (SLIP ’04)*, pp. 53–60, Paris, France, February 2004.

[35] S.-Y. Lin, C.-H. Huang, C.-H. Chao, K.-H. Huang, and A.-Y. Wu, “Traffic-balanced routing algorithm for irregular mesh-based on-chip networks,” *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1156–1168, 2008.

[36] V. de Paulo and C. Ababei, “A framework for 2.5D NoC exploration using homogeneous networks over heterogeneous floorplans,” in *Proceedings of International Conference on ReConFigurable Computing and FPGAs (ReConFig ’09)*, pp. 267–272, Cancun, Mexico, December 2009.

[37] L. Shang, L.-S. Peh, and N. K. Jha, “Dynamic voltage scaling with links for power optimization of interconnection networks,” in *Proceedings of IEEE International Symposium on High-Performance Computer Architecture (HPCA ’03)*, 2003.

[38] F. A. Samman, T. Hollstein, and M. Glesner, “Networks-on-chip based on dynamic wormhole packet identity mapping management,” *VLSI Design*, vol. 2009, Article ID 941701, 2009.

[39] P. Zipf, G. Sassatelli, N. Utlu, N. Saint-Jean, P. Benoit, and M. Glesner, “A decentralised task mapping approach for homogeneous multiprocessor Network-On-Chips,” *International Journal of Reconfigurable Computing*, vol. 2009, Article ID 453970, 14 pages, 2009.

[40] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, “Multilevel hypergraph partitioning: applications in VLSI domain,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, pp. 69–79, 1999.

[41] G. Karypis, hMetis, 2009, http://glaros.dtc.umn.edu/gkhome/metis/hmetis/download.

[42] Y. Chang, Y. Chang, G. Wu, and S. Wu, “B*-trees: a new representation for non-slicing floorplans,” in *Proceedings of the 37th Design Automation Conference (DAC ’00)*, pp. 458–463, June 2000.

[43] J. Munkres, “Algorithms for the assignment and transportation problems,” *Journal of the Society of Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[44] L. Peh and W. J. Dally, “Delay model and speculative architecture for pipelined routers,” in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA ’01)*, pp. 255–266, October 2001.

[45] A. Hansson, K. Goossens, and A. Rădulescu, “A unified approach to mapping and routing on a network-on-chip for both best-effort and guaranteed service traffic,” *VLSI Design*, vol. 2007, Article ID 68432, 2007.

[46] C. Ababei, VNOC3, 2009, http://venus.ece.ndsu.nodak.edu/~cris/software.html.

[47] S. R. Vangal, J. Howard, G. Ruhl et al., “An 80-Tile Sub-100-W TeraFLOPS processor in 65-nm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, 2008.

[48] S. Bell, B. Edwards, J. Amann et al., “TILE64™ processor: a 64-core SoC with mesh interconnect,” in *Proceedings of IEEE International Solid State Circuits Conference (ISSCC ’08)*, pp. 88–81, February 2008.

[49] K. Kim, S. Lee, J.-Y. Kim et al., “A 125GOPS 583mW network-on-chip based parallel processor with bio-inspired visual-attention engine,” in *Proceedings of IEEE International Solid State Circuits Conference (ISSCC ’08)*, pp. 305–615, February 2008.

[50] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

[51] ITRS, 2007 Edition, Interconnects, http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_Interconnect.pdf.

[52] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, “A low-overhead fault tolerance scheme for TSV-based 3D network on chip links,” in *Proceedings of International Conference on Computer-Aided Design (ICCAD ’08)*, pp. 599–602, November 2008.

[53] J. Hu and R. Marculescu, “Energy- and performance-aware mapping for regular NoC architectures,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, 2005.

[54] W. Hung, C. Addo-Ouaye, T. Theocharides, Y. Xie, N. Vijaykrishnan, and M. J. Irwin, “Thermal-aware IP visualization and placement for networks-on-chip architecture,” in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD ’04)*, pp. 430–437, October 2004.