A Framework for 2.5D NoC Exploration using Homogeneous Networks over Heterogeneous Floorplans

Vitor de Paulo, Cristinel Ababei
Electrical and Computer Engineering
North Dakota State University, Fargo ND, USA
{vitor.depaulo, cristinel.ababei}@ndsu.edu

***Abstract*—In this paper, we propose a new 2.5D NoC architecture that uses a *homogeneous* network on one layer on top of a *heterogeneous* floorplanning layer. The purpose of this approach is to exploit the benefits of compact heterogeneous floorplans and regular mesh networks through an automated design space exploration procedure. A design methodology consisting of floorplanning and router assignment in a specifically designed tool that integrates a cycle accurate NoC simulator is implemented and used to investigate the new architecture. We use this tool to compute the flit latency and compare it to a conventional 2D implementation. The separation of cores and network on two different layers offers additional area that we use to improve the network performance by searching for the optimal buffers size, number of virtual channels or mesh size. Experimental results are application specific with potential significant performance improvements for some testcases.**

***Keywords*-Homogeneous Network-on-Chip; heterogeneous floorplan; 3D circuits;**

I. Introduction

Network-On-Chip (NoC) represents a new design paradigm of complex Systems-On-Chip (SoCs) [1]. Its high scalability and predictability enable designers to design increasingly more complex systems, with large numbers of IP/cores and lower communication latency demands. In this scenario when flexibility and predictability are primary concerns, the use of a homogeneous network prevails. However, regular homogeneous NoC topologies have limitations in that communication locality is poorly supported and the utilization of network resources is low. Moreover, different-sized IP/cores do not fit well in floorplans preferred by regular network topologies. Therefore, when area and performance are more important, custom heterogeneous networks are the best choice. However, this approach suffers from poor scalability and the need for specialized packet routing algorithms.

Many previous works assumed the same area for all tiles (Fig.1.a), which simplified the design through the use of regular mesh NoC topologies. In this approach the routers are located on the same device layer of the chip where the IP/cores reside. They contribute to area increase of the floorplan that can affect negatively the yield and performance. However, assuming the same tile size becomes unrealistic in cases where SoCs integrate IP/cores of various sizes [2] (Fig.1.b). In this case, the consideration of floorplanning possibly concurrently with scheduling and mapping becomes necessary.

In this paper, we propose a new 3D NoC architecture comprised of a *homogeneous* network on one layer on top of a *heterogeneous* floorplanning layer. Because we use only two device layers we refer to it as *2.5D NoC architecture*. Our goal is to exploit the benefits of compact heterogeneous floorplans with those of a regular homogeneous mesh network through an automated procedure.

II. Related Work and Contribution

Performance benefits of 3D NoC topologies were investigated analytically in [3] and experimentally in [4]. The transition from 2D to 3D NoC architectures is done by equally distributing tiles onto the device layers forming the 3D architecture [5],[6]. Another approach is to reduce the foot print of a single tile by implementing its processing element and router in a distributed manner across all layers [7]. By leveraging long wires to connect remote intra-layer nodes, a low-diameter 3D network was reported in [8]. The first demonstration of a fabricated 3D NoC was presented in [9]. Please note that all the above works studied homogeneous NoC's.

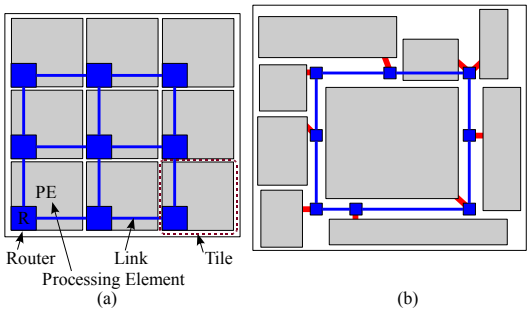

Fig. 1. (a) Homogeneous NoC design. A tile is composed of a router (R) and a generic processing element (PE). A PE can implement any IP/core of a given application. Routers are interconnected via physical links. (b) Custom heterogeneous NoC design.

Floorplanning information is used in the application specific area-wirelength calculations [10],[11] or during mapping [12]. A floorplanner is used to compute link power consumption and to detect timing violations of application-specific NoC's [13]. A design method for custom NoC topologies was presented in [14],[15]. A floorplanner is used during the topology design process, which does not consider routers (assumed to be located at the corners of IP/cores). Frequently communicating resources are placed next to each other in the irregular network growing heuristic presented in [16]. A physical planner is used in [17] during topology design to reduce power consumption on wires. The NoC synthesis approach from [13] was extended in [18] to design custom 3D network topologies. For excellent surveys on NoC in general the reader is referred to [19],[20].

In this paper we propose the use of heterogeneous floorplanning for application placement together with a two-layer 2.5D homogeneous NoC architecture. Our main contribution can be summarized as follows:

- We propose the use of a two-layer 2.5D architecture. The first layer is used entirely for IP/cores while the second layer is dedicated to the NoC implementation. 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 [21].

- We propose the use of a floorplanning and router assignment methodology for the placement of IP/cores on the first layer and their connection to the NoC on the second layer. An 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. Being on the second layer, the NoC routers have now additional available area that can be used to implement networks with more routers or enhanced routers (such as using more buffers or virtual channels).
- We designed and implemented a versatile software framework to investigate the benefits of the 2.5D architecture. It integrates an efficient B*Tree based floorplanner with a cycle accurate NoC simulator for maximum confidence in the experimental results.

III. 2.5D Architecture and Design Methodology

A. Proposed 2.5D Architecture

The architecture proposed in this paper uses two device layers. We refer to it as the *2.5D architecture*. The first layer is used entirely for the *heterogeneous* IP/cores while the second layer is dedicated to the *homogeneous* NoC. 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, on top of the first, accommodates the regular mesh network (Fig.2). In this way, the network regularity is maintained for flexibility and delay predictability while the IP/cores can have arbitrary sizes. Also, a simple packet routing algorithm can be used such as the deterministic XY routing. Another advantage of using only two layers (compared to truly 3D architectures with more device layers [6]) is that the 3D fabrication technology becomes simpler as the misalignment is only between two layers. Also, the thermal management of such a 2.5D architecture is simpler and the yield better due to the smaller footprint. The connections between IP/cores and their assigned routers are implemented using through silicon vias (TSV) between the two layers of the 2.5D architecture. Routers connected to IP/cores have five ports while the rest of the routers have only four ports.

The extra space available on the second layer can be used for additional network optimization. Routers can be designed with larger buffer space and/or more virtual channels or the mesh size can be increased. Additionally, different fault/error tolerance techniques could be included such as error correcting codes to address crosstalk issues or the extra area can simply be allocated to additional wires to increase bandwidth of physical links and therefore improve performance. The extra area can also be exploited to implement thermal monitoring and management schemes such as that presented in [22] or to implement buffers for pipelining the physical links.

An example illustrating this architecture is shown in Fig.2. This example illustrates a simple application with only three IP/cores. Hence, it requires a minimum of 2×2 mesh network.

B. Design Methodology

The methodology used for exploring the proposed 2.5D architecture is presented in Fig.3. The input to our design flow is the application represented as a communication task graph whose tasks have been mapped to *floating* IP/cores. By floating we mean the fact that their location is yet to be determined by the floorplanning step.

The implementation of our methodology can take as user input several control parameters including: the mesh size $R \times R$, the number of different floorplans to be explored N, the number of best floorplans M to be recorded in the *bests list* to be evaluated later

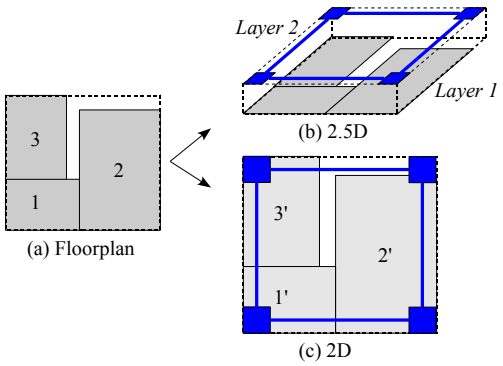

Fig. 2. (a) Original floorplan with no routers (b) 2.5D implementation (c) 2D implementation, which has every IP/core inflated to account for area required by network interfaces, routers, and wires for physical links.

using the integrated cycle accurate simulator. The main steps of the proposed methodology are as follows.

• *Exploration of N and recording of M best floorplans*

The integrated floorplanner is based on the B*Tree representation from [23]. It employs a simulated annealing algorithm, with a cost function that combines area and wirelength with different weights, which can be specified by the user. 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 M best floorplans are recorded in the *bests list*. The selection is according to the chosen criterion of smaller area or better total wirelength, which is related to the total communication volume inside the application. We bounded the search in this step to only $N = 10$ floorplans out of which only $M = 3$ are selected for detailed simulation in order to limit the total computational runtime. Ideally, one would use the router assignment and the cycle accurate simulation inside the optimization loop of the simulated annealing. However, that may become prohibitively expensive because of the long runtimes required by the cycle accurate simulator.

• *Router assignment*

Every floorplan from the list of M best floorplans will undergo an IP/core-to-router mapping step. The regular $R \times R$ mesh NoC

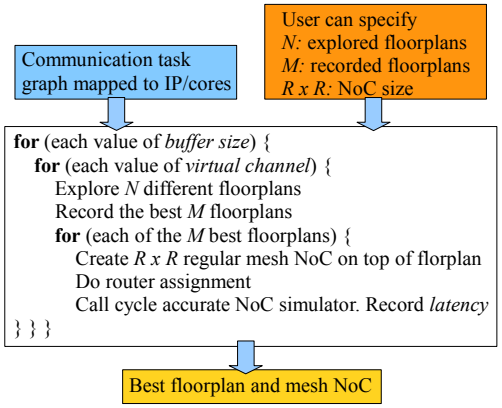

Fig. 3. Diagram of the proposed design automation methodology.

is constructed and overlapped on top of the floorplan. Initially, this square mesh grid of routers (referred to as *direct topology*) uses the minimum number of routers that can guarantee at least one router for each IP/core. However, the mesh can optionally be expanded to a higher number of routers. Because we deal with heterogeneous floorplans, it is not possible to guarantee the presence of routers at the locations of the IP/core corners (or even the IP/core layout). Therefore some of the IP/cores will have to use *extra-links* which will introduce extra delays (included inside the cycle accurate simulator) that will affect the performance.

The goal of this router assignment step is to associate every IP/core with a router from the mesh on the top layer such that the total wirelength of the extra-links between each IP/core and its assigned router is minimized. We regard this subproblem as a linear assignment problem solved using the efficient Kuhn-Munkres algorithm [24]. This algorithms uses a bipartite graph, which contains two sets of nodes: left-nodes representing the application IP/cores and right-nodes representing the routers of the mesh NoC. Edges connect every 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. A typical result after the router assignment step is shown in Fig.4.

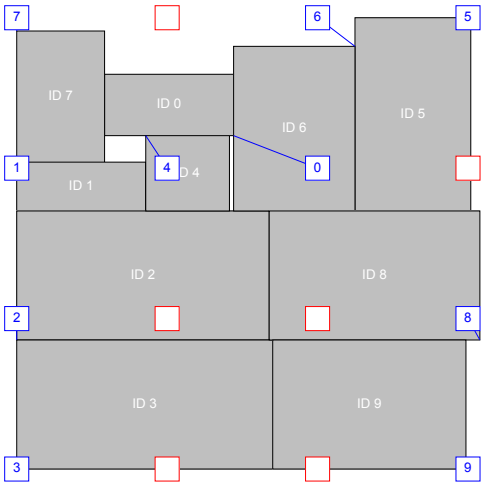

Fig. 4. Screen capture from the GUI of our tool showing the result of an IP/core-to-router mapping for *xerox*. The NoC is a direct topology of 4×4, which is the minimum to guarantee at least a router for each IP/core. Extra links are shown as straight lines between cores and assigned routers.

• *NoC simulation*

After the router assignment step, the NoC is simulated using a cycle accurate simulator integrated within our tool. The simulator is an adapted version of the one presented [22]. 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, two virtual channels. We use XY routing and wormhole flow control, which is known to be very efficient and requiring small overheads. The cycle accurate simulator is always run until all flits injected 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 [25]. The final flit latency, which is obtained during this step, is recorded for each of the floorplans from the bests list.

TABLE I
CHARACTERISTICS OF THE USED TESTCASES.

Testcase	Number of cores	Module avg. W/H	Module 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

IV. EXPERIMENTAL RESULTS

A. Experimental setup and testcases

We implemented our design methodology, which integrates the floorplanner, router assignment, NoC cycle accurate simulator and a GUI, using C++. The implementation can be downloaded from [34]. In our experiments we used five testcases whose characteristics are shown in Table I. In this table we also present the size of the *direct topology* networks with a minimum number of routers to guarantee at least a router per IP/core. We constructed these testcases from the classic MCNC testcases, whose area was scaled to achieve an average size of about $1cm \times 1cm$, which is a typical area for NoC's reported in the literature [9],[26]-[28]. The initial connectivity between the modules was used to compute the communication volume in the task graph associated with every 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 is as close as possible to one. In the NoC simulation step, each testcase had injected uniform traffic patterns as randomly-generated packets scheduled to be injected in the network at source IP/cores.

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 [22] (which was estimated for a physical link length of $1mm$ in $100nm$ technology) using a simple Elmore delay formula [29]. The same delay estimation technique was applied to the extra-links between IP/cores and routers, which were assumed to be L-shaped (with negligible via delay between metal layers). We do however consider the delay of the through silicon vias (TSV's) between the two device layers of the 2.5D architecture. We estimated the TSV delay by technology projection [30] using the delay data from [6]. Based on the arguments in [6],[31] we assume that the area required by the TSV's is negligible and that the white space available in a typical floorplan is enough to implement them.

The runtime of our tool is about 12 min (Linux machine, $2.5GHz$, $2GB$ memory) for the largest testcase *ami33* for a load sweep simulation with the injection load being varied between 20% and 80%, which translates into 70 explored floorplans out of which 21 are fully simulated using the cycle accurate simulator.

B. Exploration of the 2.5D NoC

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 specifications for minimum flit latency. We start by investigating the impact of using a larger regular mesh network ($R+k \times R+k$, $k = 1..3$) on the average flit latency. We can afford to do that because routers are expected to be smaller than the average core size, which means that there is plenty of space on the top layer that can be used for further NoC design improvements, such

as a higher number of routers or enhanced routers (bigger buffers, higher number of virtual channels, etc.).

The results for the flit latency (as obtained using the cycle accurate NoC simulator) are shown in Fig.5-9. We can see that the effect of this variation is different among testcases. For example, very good results are achieved when the mesh size was increased with 1 and 3 for *apte* (Fig.5) and *ami25* (Fig.8) respectively. However, the network performance did not improve for the other three testcases. This could be due to the fact that their modules are of similar sizes and shapes, which makes for the minimum size network to represent already a good enough distribution of routers so that the total extra links delay to be negligible. We conclude that **increasing the mesh size does not always guarantee an improvement, but it can render better results in some applications**.

For better insights, we plot in Fig.10 the normalized latency (with respect to the latency achieved using the minimum mesh) for an injection load of 60% for all testcases. We selected an injection load of 60% because this is roughly the saturation point for most testcases. We suspect that several factors affect the behavior of network latency. As the mesh size increases, a negative factor is the increased average number of hops traversed by a flit. However, at the same time the total network capacity (as total available space in all buffers inside routers) increases and also the average length of the extra-links decreases which help improve the average latency. As a result, **depending on the testcase, different mesh sizes represent optimal designs, which may be different from the direct topology**.

In another set of experiments we varied the size of the routers up to $5\times$ bigger than the default size while keeping the minimum $R \times R$ network. Because we assumed the area occupied by routers to be roughly 20% of the total cores area, we could bump-up the area of every router up to $5\times$ by increasing both the input and output buffer sizes for all ports (buffers represent most of the area inside the router architecture). Due to space limitations, we report in Fig.11 only the normalized latency (with respect to the latency achieved using the minimum buffer size) for an injection load of 60% for all testcases. We observe that the **performance does improve with the increase of the buffer size**. However, once a certain buffer size is reached (which is roughly the $3\times$ data point except for *hp*) increasing the buffer size anymore does not help. The default buffer size (12 flits) is already big enough for *ami25* and any additional buffer size increase will not have a significant impact.

In a final set of experiments, instead of increasing the buffer size we varied the number of virtual channels from 2 to 6. This experimental setup uses basically the same buffer capacities as in the previous experiment with the difference that now they are organized as more virtual channels. Again, we plot in Fig.12 only the normalized latency (with respect to the latency achieved using the minimum number of virtual channels) for an injection load of 60% versus the number of virtual channels for all testcases. **We found that the optimal number of virtual channels is different for different testcases**. This is 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 can be explained by the fact that as the network gets loaded more and more with injected packets (because the network now has increased capacity), the average amount of stalling due to arbitration inside routers may increase and negatively affect the latency.

As final comments, we note that in general latency is improved by increasing the mesh size or the buffer size at lower injection loads

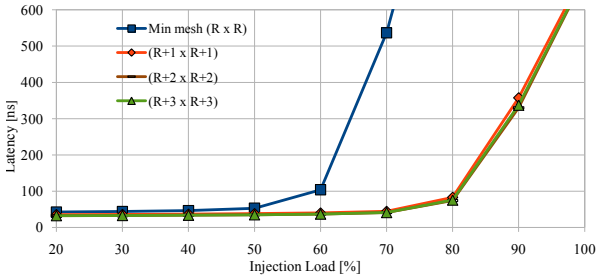

Fig. 5. Latency as the mesh size is varied for *apte*.

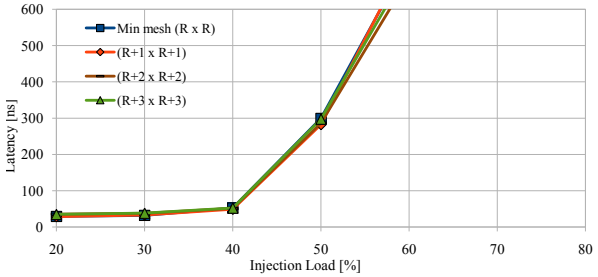

Fig. 6. Latency as the mesh size is varied for *xerox*.

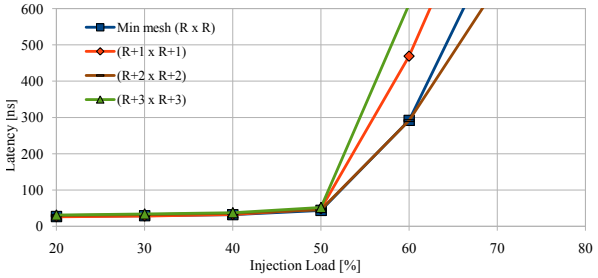

Fig. 7. Latency as the mesh size is varied for *hp*.

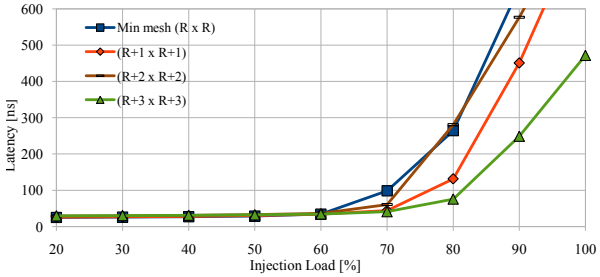

Fig. 8. Latency as the mesh size is varied for *ami25*.

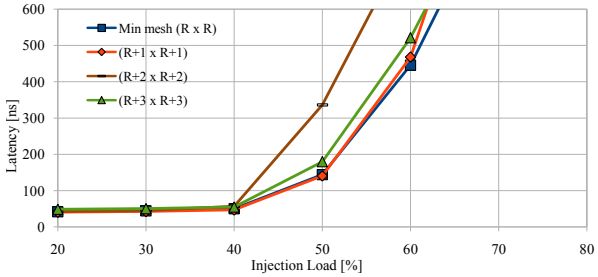

Fig. 9. Latency as the mesh size is varied for *ami33*.

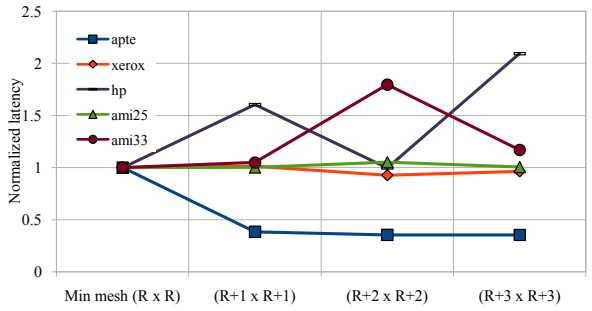

Fig. 10. Normalized latency with variation of the mesh size for 60% injection load.

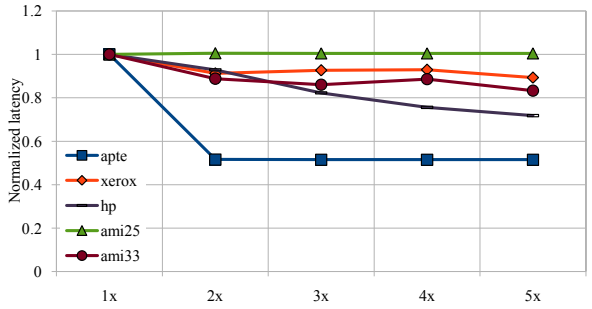

Fig. 11. Normalized latency with variation of the buffers size for 60% injection load.

and by increasing the number of virtual channels at high injection loads. Also, we found that the floorplanning step has a significant impact on the final network latency, and that increasing the number of explored floorplans can lead to better performance.

C. Comparison of 2.5D and 2D architectures

In order for us to compare the 2.5D architecture against a traditional approach we construct the 2D architecture by artificially expanding the IP/cores with a certain amount of area to account for the space taken by routers that would be implemented within the cores boundaries on the same layer. Based on the discussion in [20], we assumed a 20% extra area of the average core size among all testcases to be sufficient to implement a router with the given characteristics (size and number of buffers, number of physical ports, etc.). This is comparable to the estimation given in [5] for a five-port router using a similar technology ($90nm$ compared to our $100nm$).

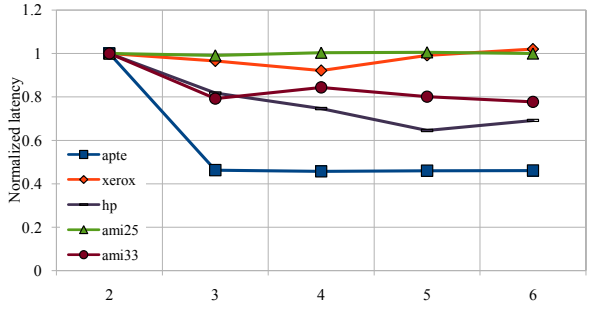

Fig. 12. Normalized latency with variation of the number of virtual channels for 60% injection load.

The simulation results are shown in Fig.13-17. In the 2.5D architecture, cores are smaller, and therefore the network links in the mesh are shorter, which should lead to smaller flit latencies when compared to the 2D situation. However, as the number of cores goes up (i.e., the mesh size) the average number of hops that a flit needs to traverse inside the network also increases. Because in the 2.5D architecture we have the TSV delay added to each extra link, as the number of cores increases we have more delay from this source too. In other words, the extra 20% area increase for the 2D achitecture translated into only slightly longer physical links of the regular 2D NoC. However, if we exploit the extra area on the top layer of the 2.5D architecture as described in the previous section, then the performance can be improved significantly in most of the testcases. This is illustrated as *2.5D best* data points in Fig.13-17.

We would like to also note that the 2D architecture performance can also be significantly improved if a custom NoC is designed similar to those presented in [13],[14]. Nevertheless, that is at the expense of scalability and predictability. However, the main goal of this paper is not to show that *regular* homogeneous 3D NoC's are better than *custom* 2D NoC's, which is unlikely, but to propose the 2.5D architecture as an alternative to *regular* 2D NoC's and explore its performance when the mesh size, buffer size and number of virtual channels are varied to use the extra space available on the top layer.

V. Conclusion and Future Work

In this paper, we proposed a new 2.5D NoC architecture. It uses a *homogeneous* network on one layer on top of a *heterogeneous* floorplanning layer. A design methodology consisting of floorplanning and router assignment in a specifically designed tool that integrates a cycle accurate NoC simulator was implemented and used to investigate the new architecture. The separation of cores and network on two different layers offers additional area that can be used for improving the network performance by searching for the optimal buffers size, number of virtual channels or mesh size.

Our work can be extended in several directions. One idea is to use three device layers with the middle layer dedicated to implementing the network and the bottom and top layers dedicated to cores. In this way the foot print of all layers is further decreased while the number of routers without an assigned core can be minimized. Also, the average length of the extra-links will decrease too. Another idea, is to integrate directly into the cost function of the floorplanning step metrics for energy or thermal profile optimization [32],[33]. Finally, the floorplanning step could be modified to consider the allocation of white space and TSV's planning under area constraints.

Acknowledgment

Support for this work was provided by the North Dakota State University.

References

[1] G. De Micheli, L. Benini, Networks on Chip, Morgan Kaufmann, 2006.
[2] J. Duato, "Managing Heterogeneity in Future NoCs," *Int. Workshop on Network on Chip Architectures*, 2008.
[3] V.F. Pavlidis, E.G. Friedman, "3-D topologies for networks-on-chip," *IEEE TVLSI*, vol. 15, nr. 10, 2007.
[4] B. Feero, P.P. Pande, "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation," *IEEE Trans. Computers*, vol. 58, no. 1, 2009.
[5] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, M. Kandemir, "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," *ACM ISCA*, 2006.
[6] J. Kim et al., "A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures," *ACM ISCA*, 2007.

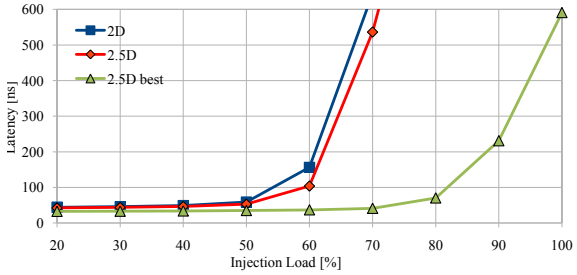

Fig. 13. Latency comparison for 2D and 2.5D implementations of *apte*.

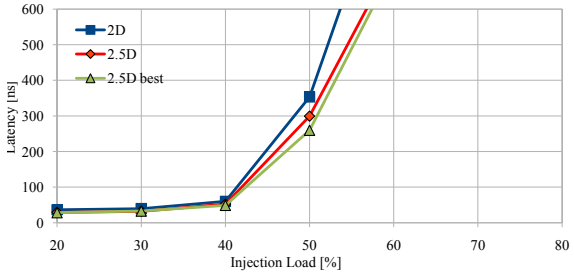

Fig. 14. Latency comparison for 2D and 2.5D implementations of *xerox*.

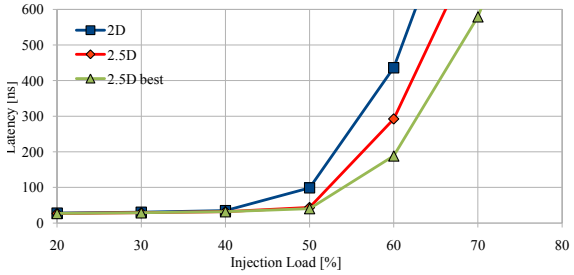

Fig. 15. Latency comparison for 2D and 2.5D implementations of *hp*.

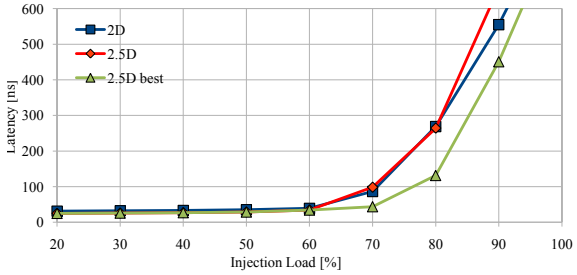

Fig. 16. Latency comparison for 2D and 2.5D implementations of *ami25*.

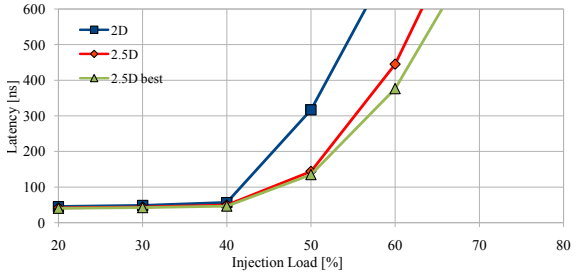

Fig. 17. Latency comparison for 2D and 2.5D implementations of *ami33*.

[7] D. Park et al., "MIRA: A Multi-layer On Chip Interconnect Router Architecture," *ACM ISCA*, 2008.

[8] Y. Xu, Y. Du, B. Zhao, X. Zhou, Y. Zhang, J. Yang:, "A low-radix and low-diameter 3D interconnection network design," *IEEE Conf. High-Performance Computer Architecture (HPCA)*, 2009.

[9] C. Mineo, R. Jenkal, S. Melamed, W.R. Davis, "Inter-Die Signaling in Three Dimensional Integrated Circuits," *IEEE Custom Integrated Circuits Conference (CICC)*, 2008.

[10] S. Murali, G. De Micheli, "SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs," *Proc. Design Automation Conference (DAC)*, June 2004.

[11] J. Xu, W. Wolf, J. Henkel, S. Chakradhar, "A design methodology for application-specific networks-on-chip," *Transactions on Embedded Computing Systems (TECS)*, vol. 5, no. 2, 2006.

[12] S. Murali, L. Benini, G. De Micheli, "Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees" *ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2005.

[13] S. Murali et al., "Designing Application-Specific Networks on Chips with Floorplan Information," *ACM ICCAD*, 2006.

[14] K. Srinivasan, K.S. Chatha, G. Konjevod, "An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks," *ACM ICCAD*, 2005.

[15] K. Srinivasan, K.S. Chatha, G. Konjevod, "Application Specific Network-on-Chip Design with Guaranteed Quality Approximation Algorithms," *ACM ASPDAC*, 2007.

[16] C. Neeb, N. Wehn, "Designing efficient irregular networks for heterogeneous systems-on-chip," *Journal of Systems Architecture - Embedded Systems Design*, vol. 54, no. 3-4, 2008.

[17] T. Ahonen, D.A. Siguenza-Tortosa, H. Bin, J. Nurmi, "Topology optimization for application-specific networks-on-chip," *Int. Workshop on System-Level Interconnect Prediction (SLIP)*, 2004.

[18] S. Murali, C. Seiculescu, L. Benini, G. De Micheli, "Synthesis of networks on chips for 3D systems on chips," *ACM ASPDAC*, 2009.

[19] T. Bjerregaard, S. Mahadevan, "A survey of research and practices of Network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, 2006.

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

[21] S.-Y. Lin et al., "Traffic-Balanced Routing Algorithm for Irregular Mesh-Based On-Chip Networks," *IEEE Trans. on Computers*, vol. 57, no. 9, 2008.

[22] L. Shang, L.-S. Peh, N.K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," *HPCA*, 2003.

[23] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, S.-W. Wu, "B*-Trees: a new representation for non-slicing floorplans," *ACM DAC*, 2000.

[24] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society of Industrial and Applied Mathematics*, vol. 5, no. 1, 1957.

[25] L.-S. Peh, W.J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," *IEEE HPCA*, 2001.

[26] S.R. Vangal et al., "An 80-tile Sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, 2007.

[27] S. Bell et al., "TILE64 - Processor: A 64-Core SoC with Mesh Interconnect," *IEEE Solid-State Circuits Conference*, 2008.

[28] K. Kim et al., "A 125GOPS 583mW Network-on-chip Based Parallel Processor with Bio-inspired Visual Attention Engine," *IEEE SSCC*, 2008.

[29] J.M. Rabaey, Digital Integrated Circuits: A Design Perspective, 2nd edition, Prentice Hall, 2003.

[30] ITRS, 2007 Edition, Interconnect, [Online]. Available: http://www.itrs.net/Links/2007ITRS/2007_Chapters/2007_Interconnect.pdf

[31] I. Loi, S. Mitra, T.H. Lee, S. Fujita, L. Benini, "A low-overhead fault tolerance scheme for TSV-based 3D network on chip links," *ACM ICCAD*, 2008.

[32] J. Hu, R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE TCAD*, vol. 24, no. 4, 2005.

[33] W. Hung, C. Addo-Quaye, T. Theocharides, Y. Xie, V. Narayanan, M.J. Irwin, "Thermal-Aware IP Virtualization and Placement for Networks-on-Chip Architecture," *Int. Conference on Computer Design (ICCD)*, 2004.

[34] C. Ababei, VNOC25, 2009, [Online]. Available: http://venus.ece.ndsu.nodak.edu/∼cris/software.html