Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2011, Article ID 405697, 6 pages
doi:10.1155/2011/405697

Research Article

Novel NoC Topology Construction for High-Performance Communications

P. Ezhumalai,[1] A. Chilambuchelvan,[2] and C. Arun[1]

[1] *Department of Computer Science and Engineering, Rajalakshmi Engineering College, Thandalam Chennai 602 105, India*
[2] *Department of Computer Science and Engineering, R.M.K Engineering College, Chennai 600 040, India*

Correspondence should be addressed to P. Ezhumalai, ezhumalai.es@gmail.com

Received 12 November 2010; Accepted 7 March 2011

Academic Editor: Youyun Xu

Different intellectual property (IP) cores, including processor and memory, are interconnected to build a typical system-on-chip (SoC) architecture. Larger SoC designs dictate the data communication to happen over the global interconnects. Network-on-Chip(NoC) architectures have been proposed as a scalable solution to the global communication challenges in nanoscale systems-on-chip (SoC) design. We proposed an idea on building customizing synthesis network—on-chip with the better flow partitioning and also considered power and area reduction as compared to the already presented regular topologies. Hence to improve the performance of SoC, first, we did a performance study of regular interconnect topologies MESH, TORUS, BFT and EBFT, we observed that the overall latency and throughput of the EBFT is better compared to other topologies, The next best in case of latency and throughput is BFT. Experimental results on a variety of NoC benchmarks showed that our synthesis results were achieved reduction in power consumption and average hop count over custom topology implementation.

1. Introduction

The integration of several heterogeneous components into a single system gives rise to new challenges. With the change of dramatic improvement in this area, it is essential to have an adaptable communication facility that can cope up with the versatile programming of the cores. Such systems will have to process data in real time, perform data transfer at the rate of hundreds of Tbps, support multiple functions and protocols for communications with standard wired and wireless interface, provide security and secrecy and cope up with time-to-market (TTM) pressures and so. Existing SoCs' communications are based on dedicated wires and shared bus (single/hierarchical) having various constraints. Dedicated wires do not provide flexibility for communication needed for hardware platforms and cannot cope up the increased number of cores. Due to exclusive access of shared bus its utilization is as low as 10%. This is inflexible to parameters required supporting heterogeneous components of SoC and not scalable. Global synchronization is not possible with this technique. It has very high signaling delay per unit length. Reduction in SNR and signal integrity and increase in cross-talk occur with the increase in parasitic capacitance and resistor as bus wire length increases. To achieve the above-mentioned requirements and to overcome the above-mentioned problems the use of a network-centric approach, that is, network-on-chip (NoC), which is globally asynchronous and locally synchronous (GALS), to integrate IPs in complex SoCs is coming to the way. Here the most important thing is here to develop the communication IP (CIP) for structured and systematic integration of heterogeneous functional IP blocks. NoC switches facilitate the routing of information from one block to another so that they may communicate with one another through message passing. The switches in this setting have buffers for each interface which queue message packets waiting for a given destination, much like in a conventional network.

The physical layout of the NoC can determine the degree of scalability and performance of the system as a whole, and just how much of an improvement over traditional bus architectures is actually attained. Discussed next are the various topological, architectural design choices on overall efficiency and performance.

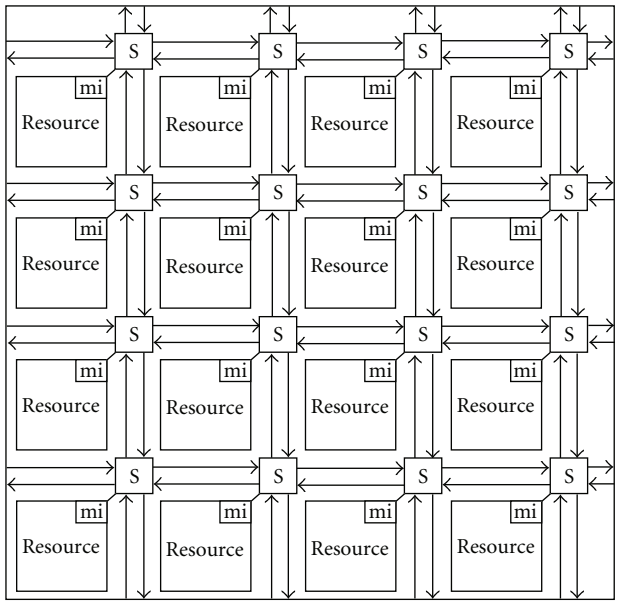


Figure 1: Mesh.

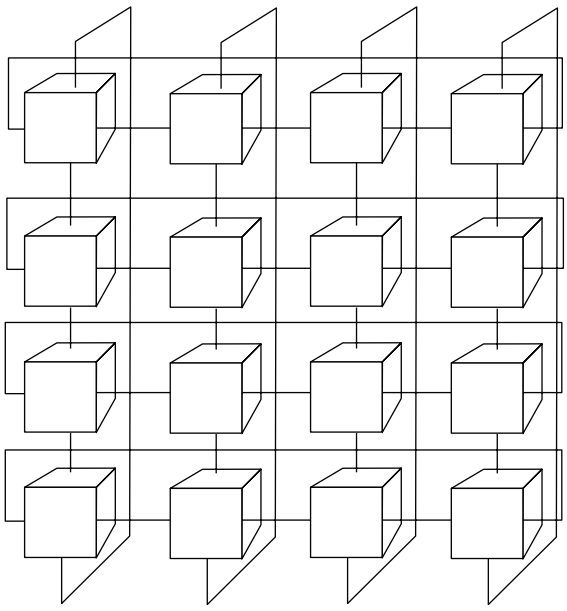


Figure 2: Torus.

2. Interconnection Architectures

2.1. Mesh Network. Every switch in mesh network Figure 1. is connected to a specific resource and the number of switches is equal to the number of resources. All switches are connected to the four closest switches and the target resource block, except those on the edge of the layout. The simplicity of such a mesh architectural layout allows for the division of the chip into processing or resource regions.

2.2. Torus Network. The torus topology in Figure 2. is similar to the mesh architecture, except that the wires are wrapped around from the top component to the bottom and rightmost to leftmost, thereby doubling the bandwidth of a mesh network. This architectural layout provides for a longer transmission distance for a given communication packet.

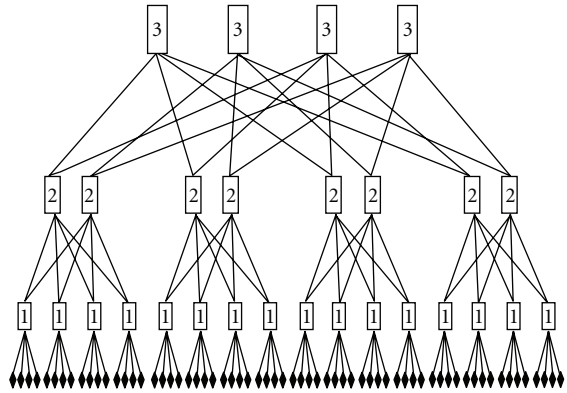


Figure 3: Butterfly fat.

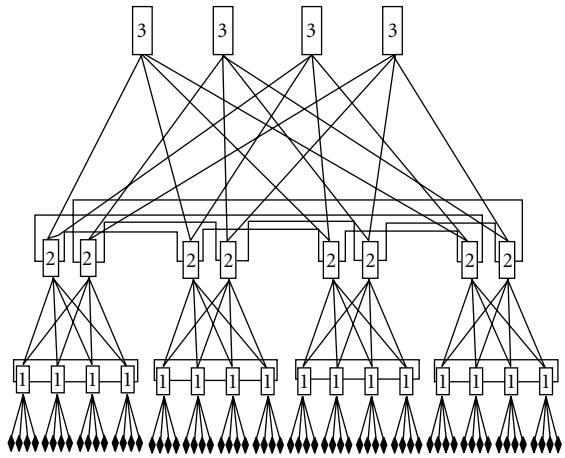


Figure 4: Extended butterfly fat.

2.3. Butterfly Fat Tree. The layout in Figure 3 is modeled in the form of a tree. Each node in the tree is represented by a set of coordinates (level, position) where level is the level in the tree and position is the spot in right-to left-ordering. Vertical levels are numbered starting at zero at the leaves. The leaves in the trees correspond to each intellectual property (IP) or component block, and the levels above represent one node for each switch, and the interconnection hierarchy maps the various connections between switches, and switches to components. Each switch is allocated two parent ports, and four child ports, or connections.

2.4. Extended Butterfly Fat Tree. The extended butterfly fat tree interconnection (EFTI) in Figure 4. is a derivative of the butterfly fat tree architecture which is the derivative of fat tree architecture The architecture uses switches of constant size. In this network too, the IPs are placed at the leaves and switches placed at the internal nodes. Each switch is allocated two parent ports, and four child ports, or connections.

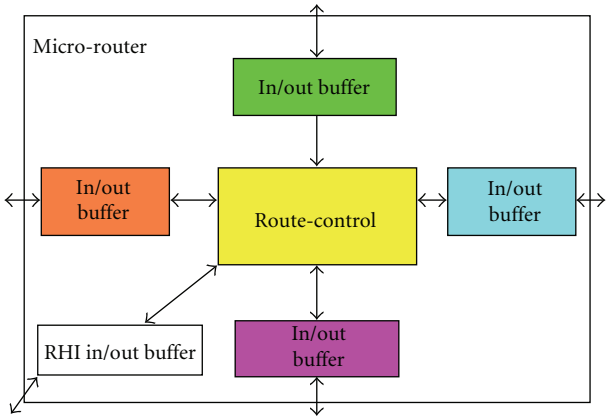


Figure 5: Proposed system architecture.

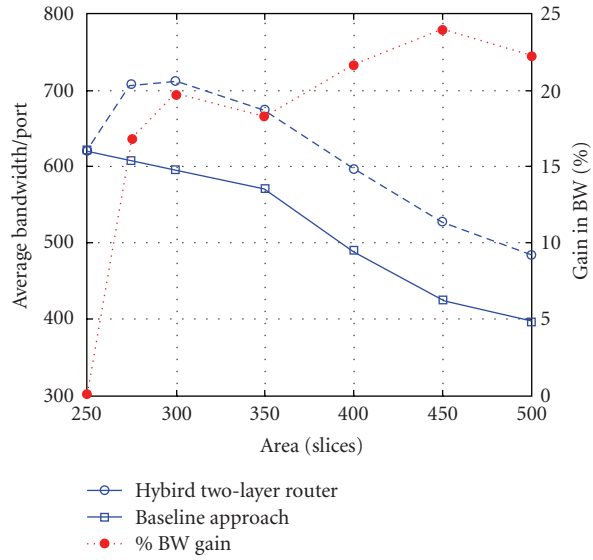


Figure 6: Area (slices) versus avg. bandwidth/port.

2.5. Existing Related Works. So far, the communication problems faced by system-on-chip were tackled by making use of regular network-on-chip architectures [1]. The following are the list of popular regular NoC architectures:

(i) mesh architecture,

(ii) torus architecture,

(iii) butterfly fat tree architecture,

(iv) extended BFT architecture.

Several existing NoC solutions have addressed the mapping problem to a regular mesh-based NoC architecture [1, 2], Ezhumalai et al. [2] proposed a survey of architectural design and analysis of network on-chip system computation of regular topologies, FPGA interconnect topologies exploration presented by Marrakchi et al. [3], custom NoC architectures with multicast routing are proposed by Yan and Lin [4] and Ezhumalai et al. [5] proposed interconnect modeling for improved system-level design optimization, long link insertion techniques for application-specific NoC architectures. Zhang et al. [6], proposed a repeated on-chip interconnect analysis and evaluation of delay, power, and bandwidth metrics under different goals presented NoC synthesis algorithms that consider system-level floor planning, but their solutions only considered solutions based on a slicing floor plan where router locations are restricted to corners of cores and links run around cores.

3. Proposed Work

Divide the problem statement into the flowing interrelated steps in Figure 5.

3.1. Input/Out Buffer. The input specification to our design flow consists of a list of modules. As observed in recent trends, many modern SoC designs combine both hard and soft modules as well as both packet-based network communications and conventional wiring. Modules can correspond to a variety of different types of intellectual property (IP) cores such as embedded microprocessors, large embedded memories, digital signal processors, graphics and multimedia processors, and security encryption engines, as well as custom hardware modules.

3.2. NoC Route Control. The portion of the input specification corresponds to the network-attached modules and their traffic flows. The nodes represent modules, edges represent traffic flows, and edge labels represent the length of the two vertices. The NoC Synthesis generates topologies based on the communication demand graph and comparing with parameters like power consumption and area usage chooses the best architecture. Below Figure 6. is an example of two architectures generated based on the given CDG.

3.3. NoC Power and Area Estimation. To evaluate the power and area of the synthesized NoC architecture, we use a state-of-the-art NoC power-performance simulator called *Orion* that can provide detailed power characteristics for different power components of a router for different input/output port configurations. It accurately considers leakage power as well as dynamic switching power, which is important since it is well known that leakage power is becoming an increasingly dominating. Orion also provides area estimates based on a state-of-the-art router microarchitecture.

3.4. Problem Formulation

3.4.1. Flow Partitioning. Flow partitioning is performed in the outer loop of our synthesis formulation to explore different partitioning of flows to separate subnet works. We make use of the following algorithm to implement flow partitioning.

3.4.2. Steiner-Tree-Based Topology Construction. For each flow partition considered, physical network topologies must be decided. In current process technologies, layout rules for implementing wires dictate physical topologies where the network links run horizontally or vertically. Thus,

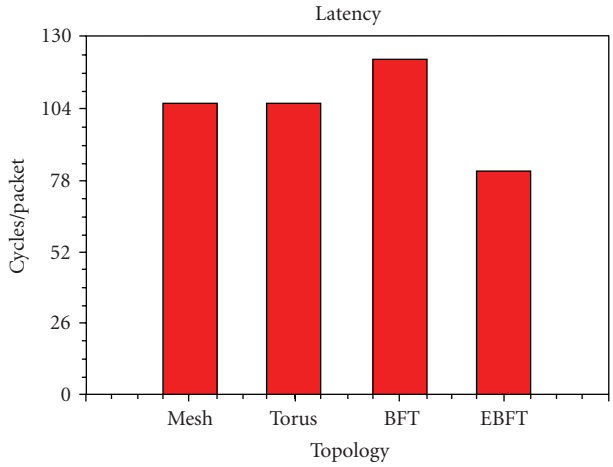


Figure 7: Latency.

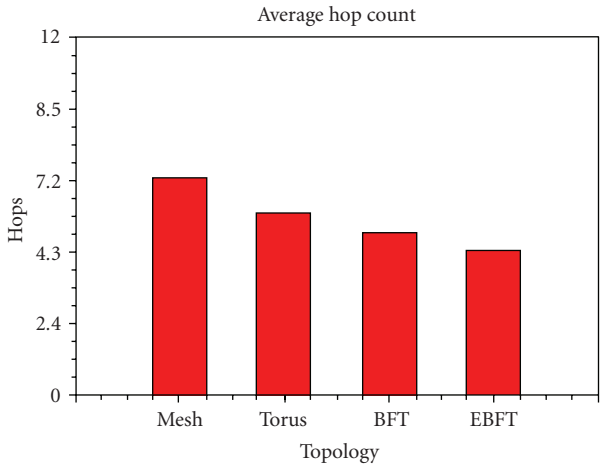
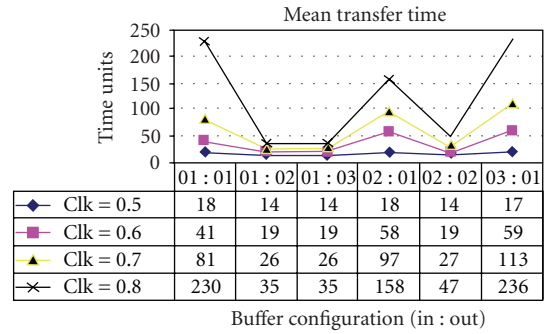


Figure 8: Throughput.

the problem is similar to rectilinear steiner tree (RST) problem that has been extensively studied for the conventional VLSI routing problem. Given a set of nodes, the RST problem is to find a network with the shortest edge lengths using horizontal and vertical edges such that all nodes are interconnected. The RST problem is well studied with very fast implementations available. We create an RST solver in the inner loop of flow partitioning to generate topologies for the set of partitions considered.

3.5. Implementation Results

3.5.1. Performance Analysis Regular NoC Topologies. We used a Gp-NoC simulator in java supporting wormhole switching and deadlock-free deterministic routing for mesh, torus, butterfly fat tree and extended-butterfly fat tree for performance analysis. The network is formed automatically depending on the number of IPs. Functional IP injected variable-size packets (mean = 200 bytes) to random destinations except



	01 : 01	01 : 02	01 : 03	02 : 01	02 : 02	03 : 01
Clk = 0.5	18	14	14	18	14	17
Clk = 0.6	41	19	19	58	19	59
Clk = 0.7	81	26	26	97	27	113
Clk = 0.8	230	35	35	158	47	236

Figure 9: Time analysis of different data length and topologies.

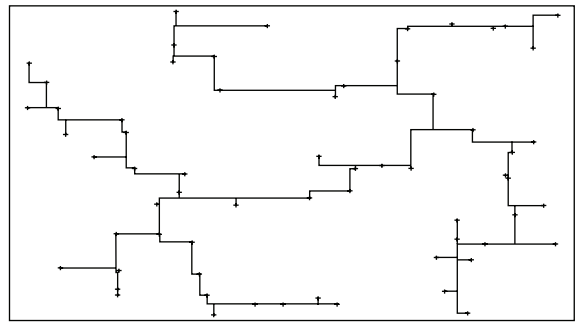


Figure 10: Steiner minimal tree: 64 points, length = 56729.

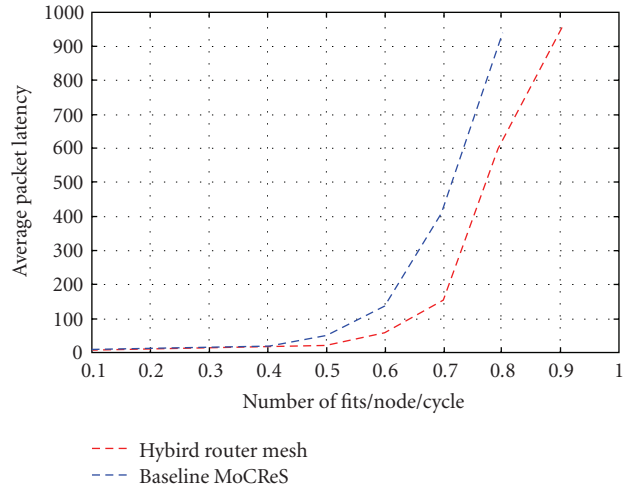


Figure 11: NOC power comparisons.

for themselves at a uniform distribution rate. The total number of IP considered for all topologies is 64; the simulation is run for 100000 cycles with asynchronous traffic type. Latency of a packet is calculated from the instant the packet's flits are presented in Figure 7, including source queuing time. Throughput is defined as the number of flits received per cycle per IP. Figure 8. shows that the performance analysis of regular topologies Throughput. Time analysis of different data length and topologies are shown in Figure 9.

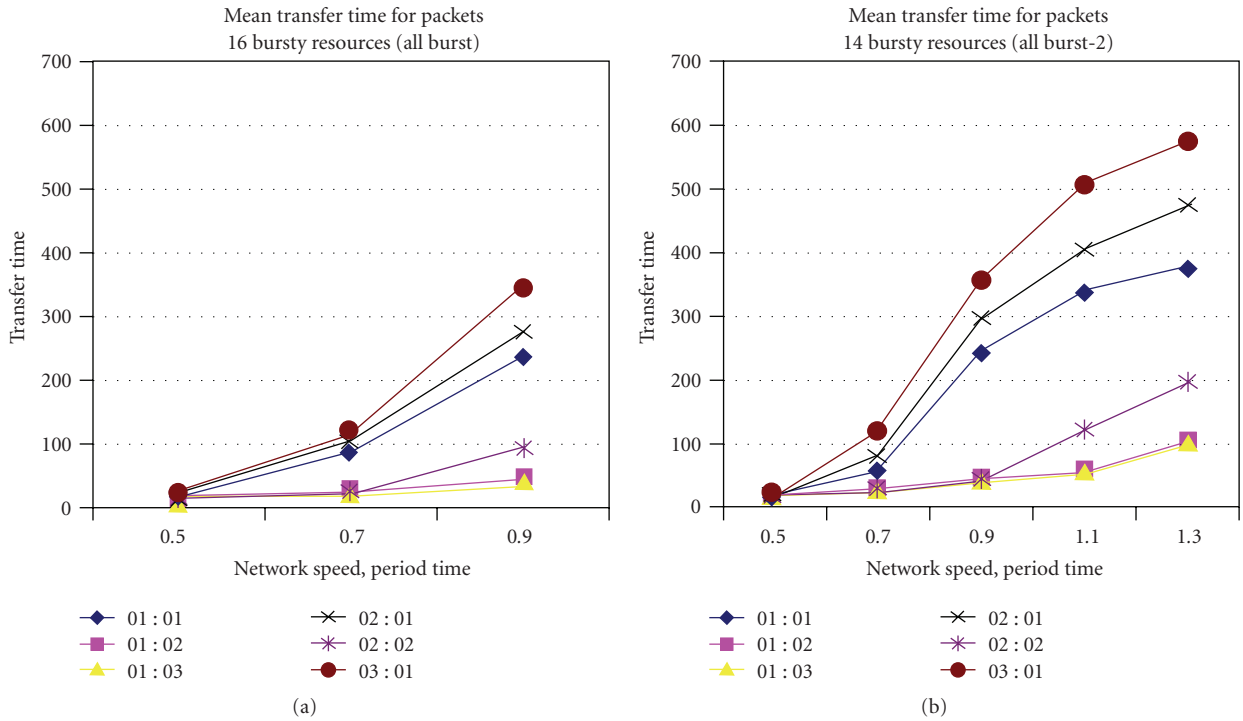


Figure 12: Transfer meantime, 16 versus 14 bursty resources.

3.5.2. Customized NoC Topology Construction. The total area as well as power consumption includes all network components. We applied the design parameters of 1 GHz clock frequency, 4-flit buffers, and 128-bit flits. For evaluation, fair direct comparison with previously published NoC synthesis results is difficult in part because of vast differences in the parameters assumed. To evaluate the effectiveness of our algorithm generates a steiner minimal tree: 64 Points, length = 56729, is shown in Figure 10, we have the full mesh implementation for each benchmark for comparison from previous published papers have been taken. These comparisons are signified to show the benefits of custom NoC architectures.

The area results, power results, the execution times, and area as well as power improvements of that algorithm are reported. The results show the algorithm can efficiently synthesize NoC architectures that minimize power and area consumption as compared with regular topologies such as mesh and optimized mesh topologies.

Thus, the two line charts in Figures 11 and 12 clearly show a reduction in power and area estimates of custom NoC with mesh and optimized mesh topologies. The power reduction is at an average of 83.43 percent and 50 percent as compared to mesh and optimized mesh topologies respectively. The area reduction is at an average of 70.95 percent as compared to optimized mesh topologies.

3.5.3. Lifetime per Mesh. The average lifetime per mesh is a measure of the stability of the mesh. The lifetime of both the shortest path tree-based mesh and minimum Steiner tree-based mesh increased as network density increased. The lifetime of the shortest path tree-based mesh was significantly better than that of the minimum Steiner tree-based mesh. Only in cases of low mobility, low network density, and smaller multicast group size was the minimum Steiner tree mesh lifetime in the range of the shortest path tree mesh. In over 90% of the cases the average lifetime for the shortest path tree mesh was at least twice the When the multicast group size (sum of the number of sources plus the number of receivers) exceeded 10, the average lifetime per shortest path mesh was more than 200 seconds (i.e., more than 20% of the 1000s simulation time). When node amount of time greater than that of the minimum Steiner mesh lifetime. mobility was 5 m/s and network density was 150 nodes, the average lifetime per shortest path mesh was almost 60% of the simulation time. As the node velocity increased, the mesh lifetime decreased as expected.

3.5.4. Number of Edges per Mesh. The shortest path mesh had a significantly larger number of edges and this could be attributed to the presence of a relatively larger number of nodes per mesh as well as to the principle of the underlying shortest path tree algorithm in determining shortest minimum hop paths without any consideration on the number of edges being introduced to the tree. Only, in conditions of low node mobility and low network density, both the meshes had similar values for the number of nodes. The number of nodes per shortest path mesh grew to 1.5–2.0 times larger than the number of nodes per Steiner tree mesh as the network density increased. As a result, mesh had at least twice the number of edges compared to the Steiner tree-based mesh. The mesh lifetime is directly related

to the number of nodes and edges per mesh. Having more intermediate nodes and edges in the mesh lends robustness to the mesh. The tradeoff is a relatively larger routing overhead. There would be several paths for every source-receiver pair and packets from the source nodes go through all these paths to the receiver nodes. Hence, there would be lot of redundant packets received along a shortest path mesh vis-à-vis a Steiner-tree-based mesh. for at least 70% of the scenarios, the shortest path

3.5.5. Conclusion and Future Work. A performance analysis of mesh, torus, BFT, and EBFT are performed and it is seen that the overall latency and throughput of the EBFT is better compared to other topologies, The next best in case of latency and throughput is BFT. The mesh and torus doesn't have any disadvantage of backbone link breaking but their overall latency and throughput are high compared to fat tree topologies. This novel idea on building customizing synthesis network-on-chip with the better flow partitioning and also considered power and area reduction as compared to the already presented regular topologies, this proposed solution framework enables the decoupling of the evaluation cost function from the exploration process, thereby enabling different user objectives and constraints to be considered. NoC architecture synthesized is not necessarily limited to just trees as Steiner tree implementations of different groups, may be connected to each other to form nontree structures.

In near future, the work of identifying the best minimized customized network-on-chip in terms of other parameters like throughput, latency, link utilization, and buffer utilization can be taken into account.

References

[1] M. A. J. Jamali and A. khademzadeh, "MinRoot and CMesh: interconnection architectures for network-on-chip systems," *World Academy of Science, Engineering and Technology*, vol. 54, pp. 354–359, 2009.

[2] P. Ezhumalai, S. Aravind, and D. Sridharan, "A survey of architectural design and analysis of network on chip systems," in *Proceedings of the International conference on Signals, Systems and Communication*, pp. 87–91, College of Engineering Guindy campus, Anna University Chennai, December 2009.

[3] Z. Marrakchi, H. Mrabet, U. Farooq, and H. Mehrez, "FPGA interconnect topologies exploration," *International Journal of Reconfigurable Computing*, vol. 2009, Article ID 259837, 13 pages, 2009.

[4] S. Yan and B. Lin, "Custom networks-on-chip architectures with multicast routing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 3, pp. 342–355, 2009.

[5] P. Ezhumalai, C. Arun, and S. Manoj Kumar, "Design and simulation network on chip of switch and buffer," *International Journal of Electronic & Communication Engineering*. In press.

[6] L. Zhang, H. Chen, B. Yao, K. Hamilton, and C. K. Cheng, "Repeated on-chip interconnect analysis and evaluation of delay, power, and bandwidth metrics under different design goals," in *Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED '07)*, pp. 251–256, March 2007.