Linear Programming based Techniques for Synthesis of Network-on-Chip Architectures

Krishnan Srinivasan, Karam S. Chatha, and Goran Konjevod,
Department of CSE, PO BOX 875406, Arizona State University,
Tempe, AZ 85287-5406.
Email: {ksrinivasan,kchatha,goran}@asu.edu

Abstract— **Network-on-chip (NoC) has been proposed as a solution for the communication challenges of System-on-chip (SoC) design in nanoscale technologies. Application specific SoC design offers the opportunity for incorporating custom NoC architectures that are more suitable for a particular application, and do not necessarily conform to regular topologies. This paper presents novel linear programming based techniques for synthesis of custom NoC architectures. The optimization objective of the techniques is to minimize the power consumption subject to the performance constraints. We present a two stage approach for solving the custom NoC synthesis problem. The physical links and routers together determine the power consumption of the NoC architecture. The power consumption of the routers is linearly dependent on the bandwidth of data flowing through them. The power consumption of the physical links is linearly dependent on both the length of the links and the supported bandwidth of data. The length of the physical links, in turn, is governed by the layout of the SoC. Therefore, in the first stage, we address the floorplanning problem that determines the locations of the various cores and the routers. In the second stage, we utilize the floorplan from the first stage to generate topology of the NoC and the routes for the various traffic traces. We present mixed integer linear programming formulations for the two stages. We also present a clustering based heuristic technique for the second stage to reduce the run times of the formulation. We analyze the quality of the results and solution times of the proposed techniques by extensive experimentation with realistic benchmarks. We also compare the custom topologies synthesized by our technique with regular mesh and QNoC based architectures.**

I. Introduction

International Technological Roadmap for Semiconductors [1] predicts that future generations of the high end SoC architectures will be implemented in less than 50 nm technology, and clocked in the $10-20\ GHz$ range. Global signal delays will span multiple clock cycles [2] [3], and make synchronous communication infeasible. Signal integrity would also suffer due to increased RLC effects. NoC has been proposed as a solution for the communication challenges in the nanoscale regime [4] [5]. NoC supports asynchronous transfer of packets. Given a suitable topology, it can provide extremely high bandwidth by distributing the propagation delay across multiple switches, and thus pipelining the signal transmission. In the lower left hand side half of Figure 1, a SoC architecture with a NoC is depicted. In the figure, the various "P/M" blocks denote processing (DSP, ASIC) cores or storage elements (SRAM, CAM), and the black boxes denote the router nodes. The lines between various blocks represent the physical links. The black blocks along with the physical links form the NoC.

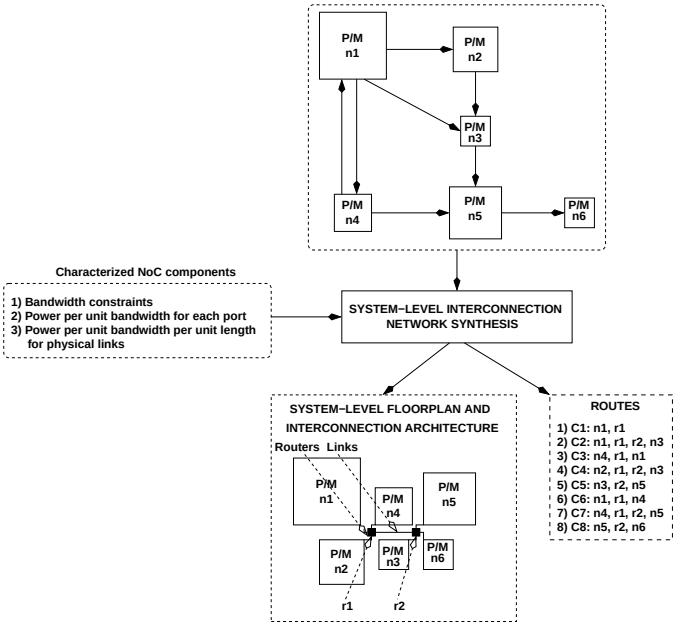


Fig. 1. System-level NoC Architecture Synthesis

This paper addresses the design of a NoC in the context of application specific SoC architecture. Application specific SoC design offers the opportunity for incorporating custom NoC architectures that are optimized for the target problem domain, and do not necessarily conform to regular topologies. Regular topologies are suitable for general purpose architectures such as the MIT RAW [6] that include homogeneous cores. Application specific SoC architectures consist of heterogeneous cores and memory elements which have vastly different sizes. For such architectures, as the results of the paper (and those by [7]) demonstrate, the custom NoC architecture is superior to regular architecture in terms of power and area consumption under identical performance requirements. Re-use of topologies and reduced design time are the two primary advantages of regular NoC architectures.

There are a number of limitations of the regular interconnection architecture. It assumes that all cores are of the same size, which is not the case for custom SoC. Therefore, even if the system-level topology is regular, it does not remain regular after the final floorplanning stage. The alternative option of regular layout results in large amount of area overhead. The regular topology assumes that every core has equal communication bandwidth with every other core. This does not hold in realistic applications. Regular

topologies are known to support design reuse. In custom topologies, the router architecture itself is regular and can be easily parameterized (on number of ports, width of physical links, number of virtual channels and so on). In other words, custom NoC architectures also support design reuse. Therefore, this paper concentrates on custom NoC topologies that are optimized for the target application. We also extend our techniques to mesh based interconnection architectures, and compare them against custom topologies.

The complexity of automated system-level design can be addressed by decomposing it into two stages: (i) computation architecture synthesis, (ii) physical design and communication architecture synthesis. The output of the computation architecture design stage would be a collection of processing and memory cores, and a mapping of the computation and storage operations on the respective cores. This problem has been addressed widely in the system-level synthesis [8] [9] [10] community. Many of these techniques do not consider a detailed communication architecture and hence, the existing techniques could be applied toward computation architecture design that utilizes NoC. This paper focuses on the problem of automated application specific NoC synthesis.

Communication architecture synthesis is shown in Figure 1. The input to the communication architecture synthesis problem is the computation architecture specification, characterized library of interconnection network components, and performance constraints. The computation architecture consists of processing and memory elements shown by rectangular blocks labeled as "P/M" in the top of the figure. Each "P/M" block is uniquely identified by a node number "n i" as denoted within each rectangle. The physical dimensions of the blocks are also specified as part of the inputs.

The directed edges between any two blocks represent the communication traces. The communication traces are annotated as "Cm(B,L)" where 'm" represents the trace number, "B" represents the bandwidth requirement, and "L" is the latency constraint. The bandwidth and latency requirements of a communication trace can be obtained by profiling the system-level specification in the context of overall application performance requirements. The traffic model that we assume for synthesis abstracts away the transaction based mechanism to derive a continuous stream model. Consider a core that sends a packet of size 256 bits every 1000 clock cycles. Assuming that the clock cycle is 3 ns, the communication trace can be abstracted as a continuous stream with 85 $Mbps$ bandwidth requirement. Thus, although the actual design would perform transaction based communication, we assume there is a continuous stream of equivalent bandwidth. Applications in multimedia and network processing domains demonstrate well defined periodic communication characteristics and hence, can be easily modelled in the trace graph.

The characterized library of interconnection architecture components is depicted on the left hand side of the figure. In nanoscale technologies, minimizing power consumption is a first order design goal along with performance maximization. Therefore, the components are characterized for both performance and power consumption. Each router architecture is characterized by i) the number of router ports, ii) the peak bandwidth supported at input or output ports, and iii) the power consumed to transfer data across the ports. The power consumption in a port is a function of the total traffic following through it. Hence, the port is characterized by power consumed per unit bandwidth of traffic. For example, the library might contain a 5-port router with an upper limit of 5.12 $Gbps$ on the bandwidth that can be supported at each port, and an power consumption of 65.6 $nW/Mbps$ and 328 $nW/Mbps$ on output and input ports, respectively. In nanoscale technologies global physical links are also significant consumers of power. The power consumption in the physical link is a function of the bandwidth of data following through the link, and the length of the link. Therefore, the physical links are characterized by power consumed per unit bandwidth per unit length. For example the physical links might be characterized as consuming 79.6 $nW/Mbps/mm$.

The output of the communication architecture synthesis problem is a system-level floorplan of the final design, topology of the network, and static routing of the communication traces on the network such that the performance constraints are satisfied, and the power consumption is minimized. Accurate estimation of power consumption due to the physical links requires estimates for link lengths. Consequently, system-level floorplanning is performed as part of the communication architecture synthesis. The topology of the network specifies the number of routers, and their interconnections. The static routing of a communication trace is shown on the right hand side of Figure 1. For example , C2 begins from "n1", passes through "r1" and "r2", and ends at "n3".

In the following sections we first discuss the architecture and characterization of the router that we utilized for generating the experimental results, and then formally define the custom NoC synthesis problem.

A. Router Architecture and power characterization

Figure 2 shows the top-level (left hand side) and detailed (right hand side) architecture of a 5-port NoC router assumed in this paper. The router consists of five unit routers that communicate with the neighboring routers, and with the processor. Unit routers inside a single router are connected through a 5x5 crossbar. Data is transferred across routers or between the processor and the corresponding router by a four phase asynchronous handshaking protocol. A single unit router, and corresponding input and output ports are highlighted in the right half of the figure. The unit router consists of input and output link controllers, input and output virtual channels, a header decoder, and an arbiter.

Data arrives at an input virtual channel of a unit router from either the previous router or the processor connected to the same router. The header decoder decodes the header flit of the packet after receiving data from the input virtual channel, decides the packet's destination direction,

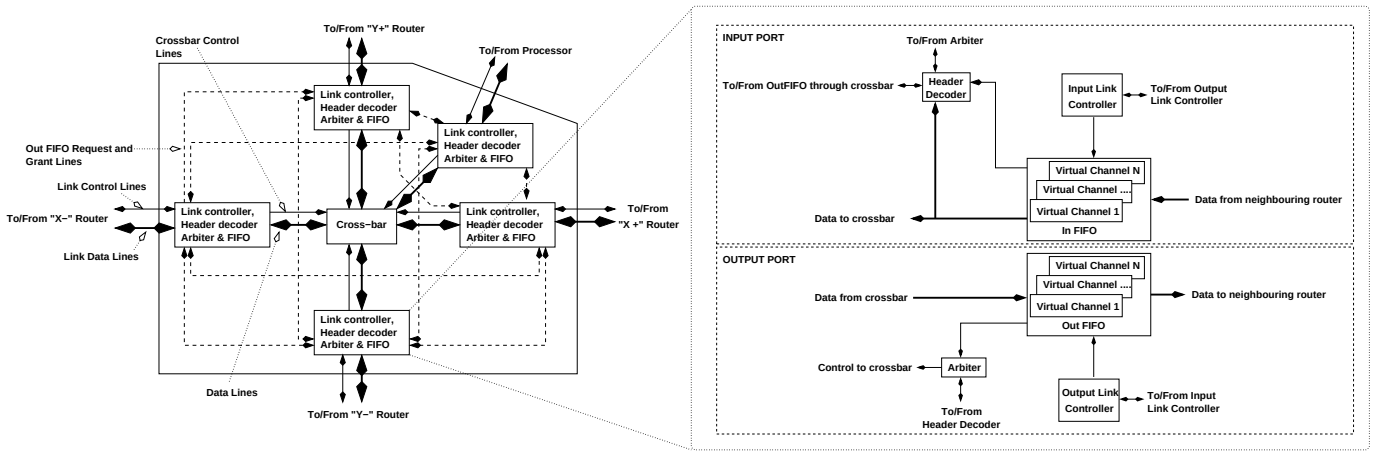


Fig. 2. Router and unit router architecture

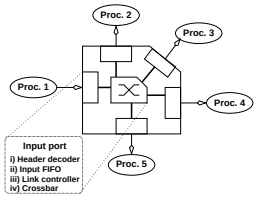


Fig. 3. Input port power consumption set-up

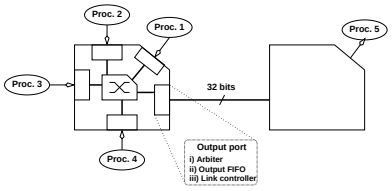


Fig. 4. Output port power consumption set-up

and sends a request to the arbiter of the unit router in the corresponding direction for access to the crossbar. In other words, the header decoder performs the task of routing the packet. We assume that the routing strategy is an application specific scheme and the header decoder decides the output port based on the communication trace identifier (ID). The communication trace ID uniquely identifies a particular data stream flowing from a source node to the destination. Once the grant is received, the header decoder starts sending data from the input to the output virtual channel through the crossbar.

We characterize the power consumption of the unit router in 100 nm technology with the help of a cycle accurate power and performance evaluator [11]. The cycle accurate simulator was constructed by characterizing the power and performance of each of the router sub-components and physical links through Hspice simulations. In the experiment the width of the physical links (consequently the width of input and output FIFO, and crossbar) is 32 bits, number of virtual channels is 2, depth of virtual channels is 4, and the number of flits in the packet is 8 (packet size 256 bits). The neighboring link controllers utilize a 2-clock cycle hand-shaking protocol to transfer a flit across the physical links. Therefore the maximum bandwidth that can be supported over the physical links is 16 bits/cycle or 0.0625 packets/cycle. The clock period was conservatively assumed to be $3ns$. The simulator was first allowed to stabilize for $3\mu s$ (1000 clock cycles), and the data was collected over the next $7\mu s$ (2333 clock cycles). Notice that the experiment is performed for the characterization of only one port of a router. Thus, 2333 clock cycles are sufficient.

A.1 Power consumption in input port of unit router

The first experiment evaluated the power consumption of a port due to the traffic flowing into the unit router. The total power consumption due to traffic at a particular input port is the summation of the power consumed by the link controller, header decoder, input FIFO and crossbar. We considered a 5-port router with 5 processors attached to each port as shown in Figure 3. Processor 1 sends packets with random contents to the 4 other processors with a uniformly random distribution. Figure 5 plots the power consumption in the input port for the 4 components (link controller, header decoder, input FIFO, crossbar) for varying injection rate at processor 1. The delay for a particular injection rate was uniformly distributed within the mean delay interval. As can be observed from the plot the power consumption in the input port varies linearly with the injection rate, and can be approximated by $P = (28 \times i)mW$ where i is the bandwidth in packets/cycle injected into the port and 28 is the slope of the line. For a clock period of $3ns$, the power per $Mbps$ of input data passing through the port is given by $P = 328nW/Mbps$.

A.2 Power consumption in output port of unit router

The second experiment evaluated the power consumed at a particular output port of a unit router due to traffic flowing in the outward direction to a neighboring router or core. We considered a 5-port router with processors (1-4) attached to four input ports as shown in Figure 4. All the four ports inject packets that traverse through the fifth port to processor 5 attached to the neighboring router. The contents of the packets where randomly generated, and the delay for a particular injection rate was also uniformly distributed within the mean delay interval. We evaluated the total power consumption of the arbiter, output FIFO, and link controller for varying cumulative injection rate from the four processors. Figure 6 plots the total power consumption (for the arbiter, output FIFO, link controller) versus the cumulative injection rate. The power consumption of the output port also varies linearly with the cumulative injection rate and can be approximated as $P = (5.6 \times i)mW$ where "i" is the bandwidth in pack-

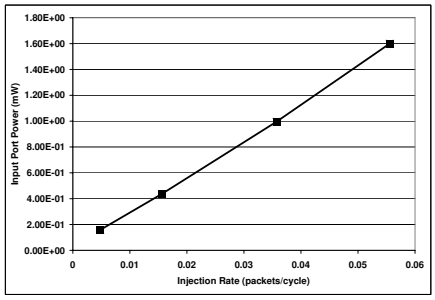


Fig. 5. Input port power consumption

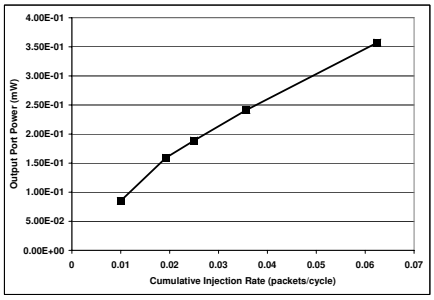


Fig. 6. Output port power consumption

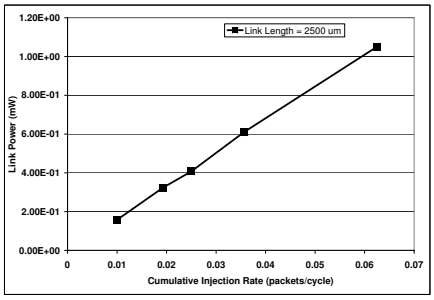


Fig. 7. Link power versus injection rate

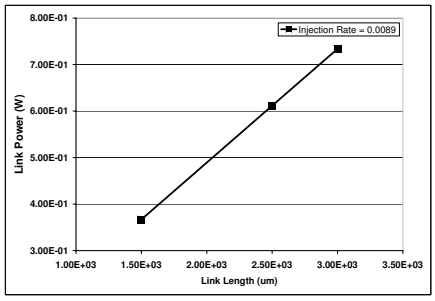


Fig. 8. Link power versus length

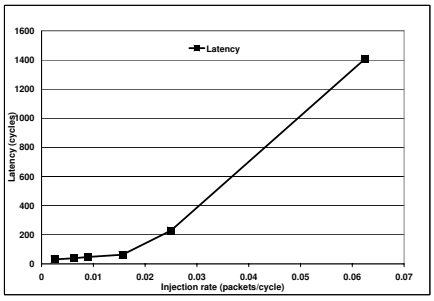


Fig. 9. Latency for 2 routers

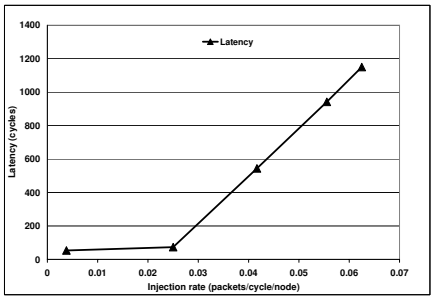


Fig. 10. Latency for 4x4 mesh

ets/cycle passing through the port and 5.6 is the slope of the line. Again, for a clock cycle of $3ns$, the power per $Mbps$ of output traffic passing through the port is given by $P = 65.5nW/Mbps$.

A.3 Power consumption in physical links

Figures 7 and 8 plot the variation in link power consumption versus injection rate (for a constant link length of 2.5 mm) and link length (for a constant injection rate 0.0089 packets/cycle), respectively. As can be observed from the figures the link power varies linearly with both injection rate and link length. The slope of the plot in Figure 7 can be approximated as $(16.915 \times i)mW$ where i is the injection rate in packets per cycle. As the link power also varies linearly with link length we can divide the slope of the plot by the corresponding length to obtain a function for the link power. Hence, the link power can be approximated as $(6.79 \times i \times l)mW$ where l is the link length in mm. Thus, with a clock cycle of $3ns$, the power consumption of the link per $Mbps$ per mm can be expressed as $P = 79.6nW/Mbps/mm$.

A.4 Latency

Figure 9 plots the average latency versus the injection rate for the packets in experiment 2 (Figure 4). Please note that the x-axis plots the injection rate due to one processor as opposed to the cumulative injection rate of the four processors as in Figure 6. As can be observed from the figure, the average latency remains constant until the output port gets congested. The injection rate at which the output port gets congested is given by 0.01563 packets/cycle/port. A similar trend is observed (see Figure 10) when we consider a 4 x 4 mesh architecture with 16 processors that are injecting to uniformly distributed random destinations. The average network latency remains constant until the network gets congested. The network congestion is marked by a sharp increase in average latency. Our synthesis technique prevents network congestion by static routing of the communication traces subject to the peak bandwidth constraint on the router ports. Since the network is always operated in the un-congested mode, we can represent the network latency constraint in terms of router hops (such as 1 or 2) instead of an absolute number (such as 100 cycles).

A.5 Other router architectures

Although the above discussion was based on a router architecture assumed by our paper, the characterization techniques, observations, and the corresponding conclusions are applicable for other router architectures as well.

B. Problem Definition

Given:

- A directed communication trace graph $G(V, E)$, where each $v_i \in V$ denotes either a processing element or a memory unit (henceforth called a node), and the directed edge $e_k = \{v_i, v_j\} \in E$ denotes a communication trace from v_i to v_j. For every $v_i \in V$, the height and width of the core is denoted by $\mathcal{H}_i$ and $\mathcal{W}_i$, respectively.
- For every $e_k = \{v_i, v_j\} \in E$, $\omega(e_k)$ denotes the bandwidth requirement in bits per cycle, and $\sigma(e_k)$ denotes the latency constraint in hops.
- A router architecture, where η denotes the number of input/output ports of the router, and Ω denotes the peak input and output bandwidth that the router can support on any one port. Thus, each port of a router can support equal bandwidth in input and output modes. Since a node $v \in V$ is attached to a port of a router, the bandwidth to

any node from a router, and from any node to a router is less than Ω. Two quantities Ψ_i and Ψ_o that denote the power consumed per $Mbps$ of traffic bandwidth flowing in the input and output direction, respectively for any port of the router.

- A physical link power model denoted by Ψ_l per $Mbps$ per mm.
- Two constants $\mathcal{H}$ and $\mathcal{W}$ that denote the height and width constraints on the overall dimensions of the system-level floorplan.
- Two constants γ_{min} and γ_{max} that denote the lower and upper bounds on the aspect ratio of the layout.

Let $\mathcal{R}$ denote the set of routers utilized in the synthesized architecture, E_r represent the set of links between two routers, and E_v represent the set of links between routers and nodes. The objective of the NoC synthesis problem is to:

- generate a system-level floorplan and
- a network topology $T(\mathcal{R}, V, E_r, E_v)$,

such that:

- for every $e_k = (v_i, v_j) \in E$, there exists a route $p = \{(v_i, r_i), (r_i, r_j), \ldots (r_k, v_j)\}$ in T that satisfies $\omega(e_k)$, and $\sigma(e_k)$,
- the bandwidth constraints on the ports of the routers are satisfied,
- the bounding box of the floorplan satisfies $\mathcal{H}$ and $\mathcal{W}$,
- the aspect ratio of the floorplan lies between γ_{min} and γ_{max}, and
- the total system-level power consumption for inter-core communication is minimized.

The NoC synthesis problem consists of the floorplanning problem, and the interconnection network generation problem. The floorplanning problem is a variation of quadratic assignment problem [12], and is known to be NP hard. The interconnection network generation problem is a variation of the generalized steiner forest problem [13], which is also known to be NP hard.

In this paper, we present a two stage approach to solving the custom NoC synthesis problem. In the first step we generate the system-level floorplan of computation architecture with an objective of minimizing the communication power consumption subject to the layout constraints. Our methodology exploits the fact that the dimensions of the routers are much smaller than those of the processing and memory cores. The possible locations of the routers are assumed to be at the corners of the various cores in the layout. In the second stage we generate the actual topology of the interconnection network and specify the routes for the various communication traces based on the floorplan generated in the previous stage.

The objective of our technique is to minimize the power consumption required for performing communication. The power consumption of the NoC is a function of the supported bandwidth of data and distance between the cores. This would imply that cores with high communication bandwidth must be placed close to each other at the expense of cores with lower communication bandwidth. However, the distance between cores is constrained by the latency of communication traces. Larger distances would imply more router hops and therefore increased latencies. In the floorplanning stage we minimize an objective function that effectively captures the trade-off between power consumption and latency.

We present linear programming based techniques for solving the system-level floorplanning problem in the context of NoC synthesis, and the actual interconnection topology generation problem. [1] We discuss optimal mixed integer linear programming (MILP) formulations for both the problems. The MILP formulation for the interconnection architecture generation in particular is constrained by large solution times. Therefore, we also present a clustering based heuristic technique for alleviating this limitation.

The solutions generated by our technique could have deadlocks between the various communication traces. However, the deadlocks can be removed by a post-processing step that introduces additional virtual channels at selected routers [14].

The paper is organized as follows: Section II discusses the previous work, Section III presents the MILP formulation, Section IV presents the clustering based approach, Section V presents the experimental results, Section VI presents a discussion on our techniques and future work, and finally Section VII concludes the paper.

II. Previous work

In recent years a number of researchers have proposed architectures, performance evaluation techniques and optimization approaches for NoC. Our work in this paper falls in the category of automated optimization approaches. In the following paragraphs, we discuss previous work in the first three categories, and then compare and contrast our work with existing research in the fourth category.

A. NoC architectures and performance models

Guerrier et al. [17], Hemani et al. [18], Sgroi et al. [19], Dally et al. [4], Benini et al. [5], and Kumar et al. [20] were early works that motivated the design of NoC for supporting on-chip communication. Taylor et al. [6] designed and fabricated a 4x4 mesh based NoC architecture as part of MIT RAW processor. In this paper, we build on the research cited above and present linear programming based techniques for automated synthesis of custom NoC architectures.

Since the early research, several researchers have proposed architectures for on-chip interconnection networks. SPIN [17] [21] [22] was one of the seminal works to propose a detailed NoC architecture built with fat tree topology. Proteo [23] [24] and xpipes [25] are router architectures that can be utilized in standard (ring, star, and bus) and arbitrary topologies, respectively. The xpipesCompiler proposed by Jalabert et al. [7] is a custom topology instantiation framework. However, it does not provide any support for NoC synthesis. Therefore, our work can be considered to be complementary to xpipesCompiler. Nostrum

[1] The paper is an integrated and extended version of our two conference papers [15][16].

[26] [27] , AEthereal [28] [29] and Vellanki et al. [30] discuss mesh based NoC architectures that support both guaranteed throughput and best effort traffic classes. Bertozzi et al. [31] and Zimmer et al. [32] presented error control schemes for on-chip buses and NoC architectures, respectively. Worm et al. [33], Chen et al. [34], Simunic et al. [35] and Nilsson et al. [36] proposed low power optimizations based on inter-router signalling scheme, buffer allocation policy, global network state, and clocking scheme, respectively.

[37] [38][39] [40] presented performance evaluation models for micro-interconnection networks that do not consider on-chip networks. Wassal et al. [41], and Ye et al. [42], proposed power models for switch fabrics of internet protocol and NoC routers, respectively. Wang et al. [43], and Bolotin et al. [44] proposed simulation based and analytical models for power evaluation of NoC. Pamunuwa et al. [45] presented analytical models for estimating the wiring overhead and the gate count for mesh-based NoC architectures. In this paper we utilized the cycle accurate power and performance model proposed by Banerjee et al. [11] to characterize the unit router and physical links, respectively.

Existing research on NoC architectures and performance models concentrate on architectures that conform to regular topologies. Application specific NoCs need not conform to a regular topology. In this paper, we present automated techniques to generate application specific irregular topologies that minimize the communication power and router requirements of the NoC.

B. Automated design techniques

In recent past, researchers have begun to address the problem of automated synthesis of NoC architectures. Pinto et al. [46] presented a technique for constraint driven communication architecture synthesis of point to point links by utilizing deterministic heuristic based k-way merging. Their technique results in network topologies that have only two routers between each source and sink. Hence, their problem formulation does not address routing. Lei et al. [47] and Ascia et al. [48] presented genetic algorithm based techniques for mapping tasks on mesh based NoC architectures. Hu et al. [49] presented a branch and bound technique to map IPs onto a regular mesh based NoC architecture. In [50], the authors extended the work presented in [49] to incorporate a deadlock free deterministic routing function. In both papers the authors assume that a NoC architecture already exists, and has a mesh topology. Our work on the other hand addresses design of an application specific NoC, and does not assume an existing interconnection network architecture. We synthesize a custom NoC architecture, and map or route the communication traces on the topology such that the performance constraints are satisfied and the communication power consumption of the NoC is minimized. In [51], the authors presented an algorithm that schedules both computation and communication transactions onto mesh based NoC architectures under real time constraints. We differ from [51] in many aspects. First, the authors address the problem of scheduling tasks on different cores that are interconnected in a NoC with regular mesh topology. We on the other hand, solve the problem of synthesis of a custom NoC topology and mapping of cores on the topology. Second, we do not assume that the traffic is transaction based. Our techniques address the synthesis and mapping of communication traces as a network flow problem.

III. MILP formulations

In this section we present the MILP formulations for solving the NoC synthesis problem. We address the problem by splitting it into two sub-problems: i) system-level floorplanning with an objective of minimizing the power consumption of the NoC subject to the layout constraints and ii) NoC topology and route generation again with an objective of minimizing the power consumption subject to the performance constraints.

A. System-level floorplanning

We present an NoC centric floorplanning formulation that minimizes communication power consumption by utilizing a unique cost function. At the floorplanning stage the power consumption due to the interconnection architecture can be abstracted as the power required to perform communication via point to point physical links between communicating cores. Although, such a cost function does not include the router power consumption, it is a true representation of the power consumption due to the physical links. However, inclusion of only the power consumption in the cost function ignores the performance requirements on the communication traces. Bandwidth constraints on the communication traces can be satisfied by finding alternative routes or adding more interconnection architecture resources. However, satisfying latency constraints is more difficult if the cores are placed wide apart. In addition to minimizing power and latency, it is also important that the layouts consume minimum area. Therefore, we specify our minimization goal as a linear combination of the power-latency function, and the area of the layout. Mathematically, we minimize

$$\alpha \cdot \left[\sum_{\forall e(u,v) \in E} dist(u,v) \cdot \Psi_l \cdot \frac{\omega(e)}{\sigma^2(e)} \right] + \beta \cdot [X_{max} + Y_{max}]$$

where $dist(u,v)$ is the distance between the cores u and v, α and β are constants, and X_{max} and Y_{max} represent the boundaries in positive X and Y directions, respectively. Note that minimizing X_{max} and Y_{max} minimizes the area that is given by $X_{max} \times Y_{max}$. The MILP is formulated such that the layout is obtained in the first quadrant. Thus, all the co-ordinates are greater than or equal to zero. The above optimization function gives a higher priority to latency constraint of a communication trace as opposed to the bandwidth. The values of the constants α and β determine the relative weight given to power minimization compared to area minimization and are specified by the designer. In the following section we describe the MILP

formulation[2].

A.1 Variables

Independent variable: As mentioned before, we assume that the cores are placed in the first quadrant of the XY plane. For each core $v_i \in V$ let $(X_{i,min}, Y_{i,min})$ denote the lower left hand side co-ordinate of the placed core.

Dependent variables: The formulation utilizes the following derived variables:

- For each core $v_i \in V$ let $(X_{i,max}, Y_{i,max})$ denote the upper right hand side corner of the placed core. Thus,

$$X_{i,max} = X_{i,min} + \mathcal{W}_i, \quad Y_{i,max} = Y_{i,min} + \mathcal{H}_i$$

- For each pair of cores $v_i, v_j \in V$, let $\mathcal{DX}_{i,j}$ denote the difference between the x co-ordinates of the top right corner of the placed cores, and $\mathcal{DY}_{i,j}$ denote the corresponding difference between the y co-ordinates. Thus:

$$\mathcal{DX}_{i,j} = X_{i,max} - X_{j,max}, \quad \mathcal{DY}_{i,j} = Y_{i,max} - Y_{j,max}$$

Since these differences can be negative, $\mathcal{DX}_{i,j}$ and $\mathcal{DY}_{i,j}$ are un-restricted variables.

- For each pair of cores $v_i, v_j \in V$, let $\mathcal{X}_{i,j}$ and $\mathcal{X}'_{i,j}$ denote binary (0,1) variables that are given by:

$$\mathcal{X}_{i,j} = \begin{cases} 1 & \text{if } X_{i,min} \geq X_{j,max} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{X}'_{i,j} = \begin{cases} 1 & \text{if } X_{j,max} > X_{i,min} \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{X}_{i,j}$ and $\mathcal{X}'_{i,j}$ can be obtained by the following linear equations:

$$X_{i,min} - X_{j,max} - \mathcal{X}_{i,j} \cdot MAXVAL < 0$$

$$X_{j,max} - X_{i,min} - \mathcal{X}'_{i,j} \cdot MAXVAL \leq 0$$

$$\mathcal{X}_{i,j} + \mathcal{X}'_{i,j} = 1$$

where $MAXVAL$ is a very large integer.

- Let $\mathcal{Y}_{i,j}$ and $\mathcal{Y}'_{i,j}$ denote similar quantities along the y co-ordinates.

A.2 Objective function

The objective function for the floorplanning stage is to:

$$\text{Minimize } (\mathcal{P} + \mathcal{A})$$

where

$$\mathcal{P} = \alpha \cdot \left(\sum_{\forall e(v_i,v_j) \in E} \Psi_l \cdot \frac{\omega(e)}{\sigma^2(e)} \cdot (|\mathcal{DX}_{i,j}| + |\mathcal{DY}_{i,j}|) \right)$$

$$\mathcal{A} = \beta \cdot [X_{max} + Y_{max}]$$

[2]Although MILP formulations for system level floorplanning have been proposed in the literature [52] [53] [54], we include our formulation so that we can discuss extensions for mesh based topologies.

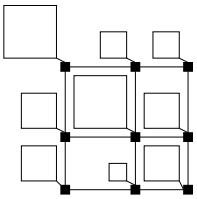


Fig. 11. Example mesh based floorplan

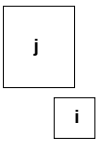


Fig. 12. Illegal layout in mesh based topology

We model $|\mathcal{DX}_{i,j}|$ by introducing two variables $\mathcal{DX}^+_{i,j}$ and $\mathcal{DX}^-_{i,j}$. We define:

$$\mathcal{DX}^+_{i,j} - \mathcal{DX}^-_{i,j} = \mathcal{DX}_{i,j} \text{ and } \mathcal{DX}^+_{i,j} + \mathcal{DX}^-_{i,j} = |\mathcal{DX}_{i,j}|$$

During minimization, the solver will set either $\mathcal{DX}^+_{i,j}$ or $\mathcal{DX}^-_{i,j}$ to zero, and the other to one. Similarly, we introduce $\mathcal{DY}^+_{i,j}$ and $\mathcal{DY}^-_{i,j}$ and define them as:

$$\mathcal{DY}^+_{i,j} - \mathcal{DY}^-_{i,j} = \mathcal{DY}_{i,j} \text{ and } \mathcal{DY}^+_{i,j} + \mathcal{DY}^-_{i,j} = |\mathcal{DY}_{i,j}|$$

A.3 Constraints

- Floorplanning requires that no two cores overlap when they are placed on the layout. Therefore, for each pair of cores $v_i, v_j \in V$ one of the following four conditions must be true:

$$X_{i,min} \geq X_{j,max}, \quad X_{j,min} \geq X_{i,max}$$

$$Y_{i,min} \geq Y_{j,max}, \quad Y_{j,min} \geq Y_{i,max}$$

Therefore,

$$\mathcal{DX}_{i,j} + \mathcal{DX}_{j,i} + \mathcal{DY}_{i,j} + \mathcal{DY}_{j,i} \geq 1$$

- Apart from minimizing the power and area, the layout should satisfy the given aspect ratio constraints. Therefore,

$$Y_{max} \geq \gamma_{min} \times X_{max}$$
$$Y_{max} \leq \gamma_{max} \times X_{max}$$

- The layout should not violate the X and Y boundaries. Therefore, for each node $i \in V$,

$$X_{i,max} \leq X_{max}$$
$$Y_{i,max} \leq Y_{max}$$

A.4 Additional constraints for mesh based topologies

We compare and contrast the custom topologies generated by our techniques against mesh based topologies. An example of the mesh based layout is shown in Figure 11. The cores in the mesh based layout are aligned along a grid. The height and width of a row and column in the grid is determined by the largest core in the particular row or column, respectively. In the mesh based floorplan, in addition to the constraints described earlier for the generic layout, we require more constraints that avoid the illegal case shown in Figure 12. In Figure 12 the two cores are not aligned along a column, and therefore cannot be laid out on a grid.

For each pair of cores $v_i, v_j \in V$ we introduce pair of binary variables $\mathcal{GX}_{i,j}$ and $\mathcal{GY}_{i,j}$ defined as:

$$\mathcal{GX}_{i,j} = \begin{cases} 1 & \text{if } X_{i,max} > X_{j,max} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{GY}_{i,j} = \begin{cases} 1 & \text{if } Y_{i,max} > Y_{j,max} \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{GX}_{i,j}$ and $\mathcal{GY}_{i,j}$ can be obtained by similar linear equations as those defined for $\mathcal{X}_{i,j}$.

The various cores will be aligned along a grid if the following equations are satisfied for each pair of cores $v_i, v_j \in V$:

$$\mathcal{GX}_{i,j} + \mathcal{X}'_{i,j} \leq 1$$

$$\mathcal{GY}_{i,j} + \mathcal{Y}'_{i,j} \leq 1$$

B. Custom interconnection topology and route generation

The power consumption of the NoC is dependent upon the length of the physical links in the architecture. We utilize the floorplan from the previous stage to select router locations, and thus determine inter router, and node to router distances. By intelligently determining router locations, we can reduce the size of the MILP formulation and thus, reduce its runtime.

Initially, we create a bounding box for each node. A bounding box is a rectangular enclosure of the node such that the bounding boxes of two adjacent nodes abut each other. For example, in Figure 13 (A), the bounding box of node 4 extends to the top boundary of node 3, and that of node 10 extends to the top boundary of node 12, and to the left boundary of node 9. In the figure, all other bounding boxes are the co-ordinates of the respective nodes.

The second step is the placement of routers. We exploit the fact that the dimensions of routers are much smaller than that of the processing cores [14]. Therefore, once the bounding boxes are generated, we place routers at the nodes of the channel intersection graph [52] formed by the bounding boxes. A channel intersection graph is a graph in which the bounding boxes form the edges, and the intersection of two perpendicular boundaries forms a node. In Figure 13 (B), the black circles depict the placement of routers at the channel intersections.

Finally, we remove all redundant routers. We remove all routers that are:

- placed along the perimeter of the layout, and
- placed less than a specified distance apart.

The motivation for removing routers can be explained as follows. The routers in the perimeter are not likely to be utilized, and are redundant. Similarly, if two routers are placed very close to each other, one of them becomes redundant as it is unlikely to be utilized in the final topology.

The location and number of routers available for the second stage depends on the algorithm used to remove redundant routers. Our algorithm for removing routers is shown in Figure 14. First, the algorithm removes the routers along the perimeter of the floorplan. Figure 13 (C) depicts the

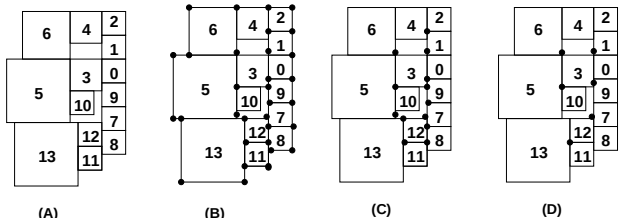


Fig. 13. Example of router allocation for custom topology

```
rem_redundant_rtrs()
1   rem_perimeter_rtrs()
2   initialize()
3   for (r ∈ R)
4       L(r) = get_close_rtr(r)
5   end for
6   cur_rtr = bottom_left_rtr()
7   rem_close_rtrs(cur_rtr)
8  return
```

```
rem_close_rtrs(cur_rtr)
1   for (r ∈ L(cur_rtr))
2     set(r) = removed
3   end for
4   set(cur_rtr) = tagged
5   next_rtr = get_next_rtr()
6   if (next_rtr = NULL)
7     return
8   else
9     rem_close_rtrs(next_rtr)
10    end if
11 return
```

Fig. 14. Algorithm for removing redundant routers

stage when the routers along the perimeter are removed. After removing routers along the perimeter, the algorithm calls the initialization function that marks all the internal routers as free. In lines 3-5, the algorithm generates a list L for each router that specifies the routers that are less than the minimum distance from it. Line 6 of the algorithm sets the current router (cur_rtr) to be the router at the bottom left hand corner of the layout, which is the router at the bottom left hand corner of node 12 in Figure 13. The next line calls the function rem_close_rtrs that removes all routers that are in $L(cur\_rtr)$, marks cur_rtr as tagged, and hops to the closest available router ($next\_rtr$) that is free (neither tagged nor removed). The function rem_close_rtrs recursively calls itself with $next\_rtr$ as parameter, until all routers are either tagged or removed. 13 (D) depicts the stage when routers are placed at more than a specified minimum distance apart.

Let $\mathcal{R}$ denote the total number of available routers. In the algorithm, each router is either tagged, or removed. A router cannot be both tagged and be removed. Therefore, the complexity of the algorithm is given by $O(|\mathcal{R}|)$.

We can further minimize the size of the formulation by limiting the number of routers that a node can be mapped to. Since the layout places communicating nodes close to each other, it is very unlikely that an optimal solution will have a node that is mapped to a router located at a large distance away from it. Therefore, for each node, we consider only those routers that are within a certain maximum distance from it. The distance is specified by the designer. Let $\mathcal{R}_i$ denote the set of routers available to node v_i.

Since we know the location of the routers, we can determine the shortest distance from a node v_i to the routers in $\mathcal{R}_i$. By the same argument, we can also determine all inter router distances.

The objective function of the formulation is minimiza-

tion of the communication power. The power consumption in the NoC is the sum of the power consumed by the routers and the physical links. The power consumed by the routers is given by the product of the bandwidth of data flowing through the ports and the characterization function that specifies the power consumption per unit bandwidth. Similarly, the power consumption in a physical link is a product of the bandwidth of data flowing through the link, length of the link and the characterization function that specifies the power consumption per unit bandwidth per unit length.

In Section III-B.1 we define the variables of the formulation, in Section III-B.2 we state the objective function, and in Section III-B.3, we present the constraints.

B.1 Variables

Base Variables: We define the following base (independent) variables.

- *Number of routers:* Let $r_i \in \mathcal{R},\ 0 \leq i \leq R_{max}$ denote a router. Each router in the NoC architecture is identical with the same number of ports "η", and peak bandwidth "Ω" per port. All ports are bidirectional.
- *Ports of the router:* Let $p_{i,j}$, $0 \leq j < \eta$, represent the j^{th} port of a router $r_i \in \mathcal{R}$.
- *Node-to-port mapping variables:* For each node $v_k \in V$, let $\mathcal{R}_k$ denote the set of routers that it can be mapped to. Let $\mathcal{NR}_{k,i,j}$ be a {0,1} variable that is 1 if node v_k is mapped to port $p_{i,j}$ of router $r_i \in \mathcal{R}_k$, otherwise 0. For each router $r_m \notin \mathcal{R}_k$, $\forall p_{m,j}, \mathcal{NR}_{k,m,j} = 0$
- *Port-to-port mapping variables:* For each port $p_{i,j}$ of router $r_i \in \mathcal{R}$, let $\mathcal{RR}_{i,j,k,l}$ be a {0,1} variable that is 1 if port $p_{i,j}$ of router $r_i \in \mathcal{R}$ is linked to port $p_{k,l}$ $(k \neq i)$ of router $r_k \in \mathcal{R}$, otherwise 0.
- *Variable for flow of traffic out of a port:* For each edge $\{v_i, v_j\} \in E$, let $\mathcal{O}_{i,j,k,l}$ be a {0,1} variable that is 1 if traffic from node v_i to node v_j flows out of port $p_{k,l}$, otherwise 0.
- *Variable for flow of traffic into a port:* For each edge $\{v_i, v_j\} \in E$, let $\mathcal{I}_{i,j,k,l}$ be a {0,1} variable that is 1 if traffic from node v_i to node v_j flows into port $p_{k,l}$, otherwise 0.

Variables $\mathcal{O}$ and $\mathcal{I}$ are utilized for modeling and satisfying the bandwidth and latency constraints on the various communication traces.

Derived Variables: We define the following derived variables.

- *Variable for the total traffic flowing out of a port:* Let $\mathcal{BO}_{k,l}$ be a variable that represents the total traffic flowing out of port $p_{k,l}$. $\mathcal{BO}$ can be derived as follows.

$$\mathcal{BO}_{k,l} = \sum_{\forall e_m = \{v_i, v_j\} \in E} \omega(e_m) * \mathcal{O}_{i,j,k,l}$$

- *Variable for the total traffic flowing into a port:* Let $\mathcal{BI}_{k,l}$ be a variable that represents the total traffic flowing into port $p_{k,l}$. $\mathcal{BI}$ can be derived as follows.

$$\mathcal{BI}_{k,l} = \sum_{\forall e_m = \{v_i, v_j\} \in E} \omega(e_m) * \mathcal{I}_{i,j,k,l}$$

- *Variable for flow of traffic on a link:* Let $\mathcal{Z}_{i,j,k,l,m,n}$ be a $\{0, 1\}$ variable that is 1 if traffic (i, j) leaves port l of router k, and port l of router k is connected to port n of router m. Hence, $\mathcal{Z}_{i,j,k,l,m,n}$ can be represented as

$$\mathcal{Z}_{i,j,k,l,m,n} = \mathcal{O}_{i,j,k,l} \times \mathcal{RR}_{k,l,m,n}$$

The non-linear equation can be easily linearized by the following rule.

$$\mathcal{O}_{i,j,k,l} + \mathcal{RR}_{k,l,m,n} \geq 2 \times \mathcal{Z}_{i,j,k,l,m,n}$$
$$\mathcal{O}_{i,j,k,l} + \mathcal{RR}_{k,l,m,n} \leq \mathcal{Z}_{i,j,k,l,m,n} + 1$$

B.2 Objective Function

The objective is to minimize the power consumption of the NoC due to the cumulative traffic flowing through input and output ports, respectively of all the routers. The objective function can be expressed mathematically as follows:

$$Minimize\ (\mathcal{P}_R + \mathcal{P}_L)$$

where

$$\mathcal{P}_R = \Psi_i \cdot \sum_{\forall r_i \in \mathcal{R}} \sum_{\forall p_{i,j}} \mathcal{BI}_{i,j} + \Psi_o \cdot \sum_{\forall r_i \in \mathcal{R}} \sum_{\forall p_{i,j}} \mathcal{BO}_{i,j}$$

$$\mathcal{P}_L = \Psi_L \left(\sum_{i,j,k,l,m,n} \omega(i,j) \cdot \mathcal{RD}_{k,m} \cdot \mathcal{Z}_{i,j,k,l,m,n} + \sum_{i,j,k,l} \mathcal{ND}_{i,k} \cdot \omega(i,j) \cdot \mathcal{NR}_{i,k,l} + \sum_{i,j,k,l} \mathcal{ND}_{j,k} \cdot \omega(i,j) \cdot \mathcal{NR}_{j,k,l} \right)$$

where Ψ_i and Ψ_o are weights that denote the power consumed per $Mbps$ of traffic flowing in the input and output directions, respectively, for any port of a router in the NoC, and Ψ_L is the link power per unit length per Mbps, $\mathcal{RD}_{k,m}$ denotes the distance between routers k and m, and $\mathcal{ND}_{i,k}$ denotes the distance of node i from router k.

B.3 Constraints

The following constraints are formulated.

- *Port capacity constraint:* The bandwidth usage of an input or output port should not exceed its capacity. Therefore,

$$\forall i \in \mathcal{R}, \forall p_{i,j}, \mathcal{BI}_{i,j} \leq \Omega, \mathcal{BO}_{i,j} \leq \Omega$$

- *Port-to-port mapping constraint:* A port can be mapped to one node, or to any one port that belongs to a different router:

$$\forall p_{i,j}, \sum_{\forall r_k \in \mathcal{R}, k \neq i} \sum_{\forall p_{k,l}} \mathcal{RR}_{k,l,i,j} + \sum_{\forall v_m \in V} \mathcal{NR}_{m,i,j} \leq 1$$
$$\forall p_{i,j}, \forall r_k \in \mathcal{R}, k \neq i, \mathcal{RR}_{k,l,i,j} = \mathcal{RR}_{i,j,k,l}$$

The first constraint above is an inequality because it is possible that a port may not be mapped to any other port or node. The second equation models the symmetry of the variable $\mathcal{RR}$.

• *Node-to-port mapping constraint:* A node should be mapped exactly to one port. Therefore,

$$\forall v_i \in V, \sum_{\forall r_k \in \mathcal{R}_i} \sum_{\forall p_{k,l}} \mathcal{NR}_{i,k,l} = 1$$

• *Traffic routing constraints:* The traffic routing constraints discussed below ensure that for every $e_k = (v_i, v_j) \in E$, there exists a path $p = \{(v_i, r_i), (r_i, r_j), \ldots (r_k, v_j)\}$ in T.

1. If a node is mapped to a port of a router, all traffic emanating from that node has to enter that port. Similarly, all traffic terminating at that node should leave from that port. Thus, for each router r_k, $\forall p_{k,l}$, and $\forall (v_i, v_j) \in E$, we require

$$\mathcal{I}_{i,j,k,l} \geq \mathcal{NR}_{i,k,l},\ \mathcal{O}_{i,j,k,l} \geq \mathcal{NR}_{j,k,l}$$

2. If a node is mapped to a port of a router, no traffic from any other node can either enter or leave that port. Thus, $\forall \{v_i, v_j\} \in E, \forall v_m \in V, m \neq i, m \neq j, \forall p_{k,l}$:

$$\mathcal{NR}_{m,k,l} + \mathcal{I}_{i,j,k,l} \leq 1,\ \ \mathcal{NR}_{m,k,l} + \mathcal{O}_{i,j,k,l} \leq 1$$

3. If a traffic enters a port of the router, it should not enter from any other port of that router. Similarly, if a traffic leaves a port of a router, it should not leave from any other port of that router. This constraint ensures that the traffic does not get split across multiple ports. Thus, for each router r_k, and $\forall (v_i, v_j) \in E$,

$$\sum_{\forall p_{k,l}} \mathcal{I}_{i,j,k,l} \leq 1,\ \sum_{\forall p_{k,l}} \mathcal{O}_{i,j,k,l} \leq 1$$

4. If a traffic enters a port of a router, it has to leave from exactly one of the other ports of that router. In the same way, if a traffic leaves a port of a router, it must have entered from exactly one of the other ports of that router. This constraint ensures the conservation of flow of traffic. Hence, for each router r_k, $\forall p_{k,l}$, and $\forall (v_i, v_j) \in E$,

$$\sum_{\forall p_{k,m}, m \neq l} \mathcal{O}_{i,j,k,m} \geq \mathcal{I}_{i,j,k,l}$$
$$\sum_{\forall p_{k,m}, m \neq l} \mathcal{I}_{i,j,k,m} \geq \mathcal{O}_{i,j,k,l}$$

5. If two ports of different routers are connected, traffic leaving from one port should enter the other, and vice versa. For example, if $p_{k,l}$ and $p_{m,n}$ are connected, $\mathcal{RR}_{k,l,m,n}$ will be 1. Therefore, a traffic $O_{i,j,m,n}$ leaving port n of router r_m should enter port l of router r_k. Therefore, $I_{i,j,k,l}$ should be set to 1. Similarly, if $I_{i,j,k,l} = 1$, $O_{i,j,m,n}$ should be set to 1. Therefore, for each pair of routers $\{r_k, r_m\}$, $k \neq m$, $\forall p_{k,l}, \forall p_{m,n}$ and, $\forall (v_i, v_j) \in E$,

$$\mathcal{RR}_{k,l,m,n} + \mathcal{I}_{i,j,k,l} - \mathcal{O}_{i,j,m,n} - 1 \leq 0$$
$$\mathcal{RR}_{k,l,m,n} - \mathcal{I}_{i,j,k,l} + \mathcal{O}_{i,j,m,n} - 1 \leq 0$$

6. If two ports of different routers are connected, a traffic can leave exactly one of the two ports. Similarly, a traffic can enter only one of the two ports. For example, if $p_{k,l}$ and $p_{m,n}$ are connected, for any traffic $(v_i, v_j) \in E$, $I_{i,j,m,n}$ and $I_{i,j,k,l}$ cannot be simultaneously 1. Similarly, $O_{i,j,m,n}$ and $O_{i,j,k,l}$ cannot be simultaneously 1. Thus, for each pair of routers $\{r_k, r_m\}$, $k \neq m$, $\forall (v_i, v_j) \in E, \forall p_{k,l}, \forall p_{m,n}$

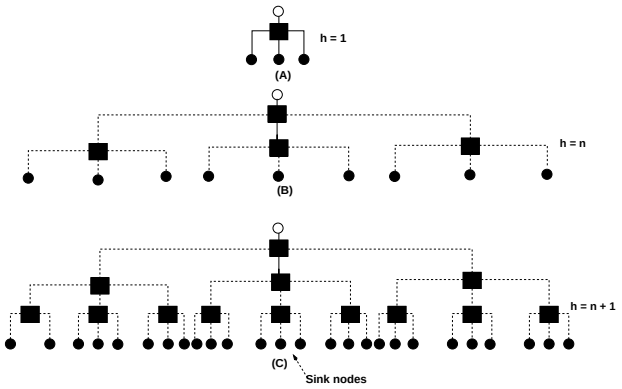


Fig. 15. Adding router to increase number of traffic

$$\mathcal{RR}_{k,l,m,n} + \mathcal{I}_{i,j,k,l} + \mathcal{I}_{i,j,m,n} - 2 \leq 0$$
$$\mathcal{RR}_{k,l,m,n} + \mathcal{O}_{i,j,k,l} + \mathcal{O}_{i,j,m,n} - 2 \leq 0$$

7. If a traffic enters a port of a router, that port must be mapped to a node or to a port of a different router. Therefore, if $\mathcal{I}_{i,j,k,l}$ is 1 for some traffic $(v_i, v_j) \in E$, some $\mathcal{NR}_{i,k,l}$ should be 1 or, some $\mathcal{RR}_{m,n,k,l}$ should be 1 where $p_{m,n}$ exists. Similarly, if a traffic leaves a port of a router, that port must be mapped to a node, or to a port of a different router. Therefore, if $\mathcal{O}_{j,i,k,l}$ is 1 for some traffic $\{v_j, v_i\} \in E$, some $\mathcal{NR}_{i,k,l}$ should be 1 or, some $\mathcal{RR}_{m,n,k,l}$ should be 1 where $p_{m,n}$ exists. The constraints can be modeled as follows. For each router r_k, $\forall p_{k,l}$, and $\forall \{v_j, v_i\} \in E$

$$\mathcal{NR}_{j,k,l} + \sum_{\forall r_m \in \mathcal{R}} \sum_{\forall p_{m,n}} \mathcal{RR}_{k,l,m,n} \geq \mathcal{I}_{j,i,k,l}$$
$$\mathcal{NR}_{i,k,l} + \sum_{\forall r_m \in \mathcal{R}} \sum_{\forall p_{m,n}} \mathcal{RR}_{k,l,m,n} \geq \mathcal{O}_{j,i,k,l}$$

• *Latency constraint:* The latency constraint refers to the maximum number of hops that is allowed to route the traffic from a source node to a sink node. For example, a latency of 2 means that the traffic can pass through at most two routers. The latency constraint is modelled as follows:

$$\forall e_k = \{v_i, v_j\} \in E, \sum_{\forall r_k \mathcal{R}} \sum_{\forall p_{k,l}} \mathcal{O}_{i,j,k,l} \leq \sigma(e_k)$$

Latency constraint of 1 is a special case in which no router to router connections are allowed. Therefore, for latency constraint of 1, all previous constraints pertaining to router to router connections can be removed. The imposition of latency constraint affects the feasibility of a NoC architecture. Latency and the number of ports in the router architecture are related by the following lemma.

Lemma: If the router architecture has η ports per router, and σ is the maximum latency constraint on any edge, (i.e $\forall e_k \in E, \sigma(e_k) \leq \sigma$), a NoC topology is not possible if for

any node, the total number of edges entering and leaving the node is more than $(\eta - 1)^{\sigma}$.

Proof: Without loss of generality, we will prove the lemma for multiple traffic traces originating from one source node, and ending at multiple respective sink nodes. As shown in Figure 15, the source node is denoted by an unfilled circle, the routers are denoted by filled square boxes, and the sink nodes are denoted by filled circles that form the leaves of the tree. We will prove our lemma by mathematical induction on σ. For simplicity, the proof assumes that bandwidth constraints are not violated.

Base case: Let $\sigma = 1$. In this case, all sink nodes have to be mapped to ports of the router to which the source node is mapped. As shown in Figure 15(A), the resulting architecture can be visualized as a η-ary tree with height 1. Since one port is taken by the source node, the maximum number of ports available for sink nodes is given by:

$$numtraces_1 = \eta - 1$$

Induction hypothesis: Suppose the assumption holds for $\sigma = \sigma_n$. Therefore, the maximum number of traces is given by:

$$numtraces_n = (\eta - 1)^{\sigma_n}$$

The resulting architecture is shown in Figure 15(B). The architecture is an η-ary tree of height σ_n, and the maximum number of traces is given by the number of leaf nodes in the tree ($(\eta - 1)^{\sigma_n}$).

Proof for $\sigma_n + 1$: An η-ary tree with height $\sigma_n + 1$ can be formed from an η-ary tree with height σ_n by introducing a router at each leaf node. As shown in Figure 15(C), introduction of a router at a leaf node is equivalent to routing traffic from source to the various sink nodes in $\sigma_n + 1$ hops. In a tree with height σ_n, each leaf node can be replaced by a router, thus increasing the number of leaf nodes in a tree with height $\sigma_n + 1$ by $\eta - 1$. Therefore, when all leaf nodes in the σ_n-height tree are replaced by routers, the maximum number of traces that can be mapped in the $\sigma_n + 1$ tree is given by:

$$numtraces_{n+1} = (\eta - 1) * (\eta - 1)^{\sigma_n} = (\eta - 1)^{\sigma_n + 1}$$

Q.E.D

IV. Clustering based heuristic technique

The MILP formulation for interconnection topology and route generation is constrained by exponentially increasing solution times for large communication trace graphs. This section presents a clustering based heuristic technique for reducing the solution times. The clustering based heuristic technique is executed after the layout has been generated. The overall approach is shown in Figure 16.

The first stage is to form clusters of nodes. The sizes of the clusters are constrained by the maximum number of nodes in the clusters. This information is specified by the designer. We utilize an algorithm by Johnson et al. [55] [56] to form our clusters. For each edge $e \in E$, the clustering algorithm assigns a distance metric to the edge given by

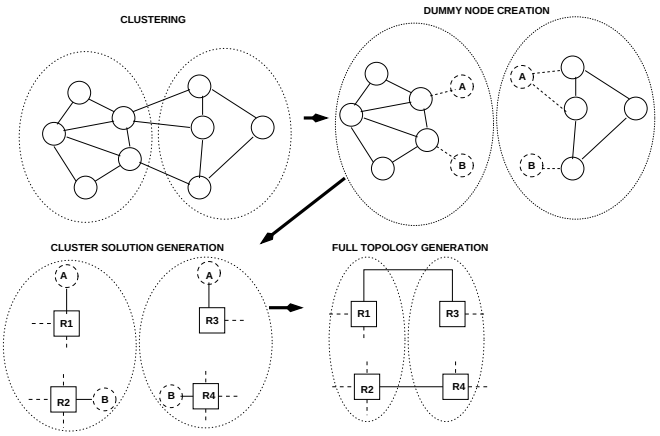


Fig. 16. Clustering based approach

$$\mathcal{DF}_e = \frac{\sigma_e^2}{\omega_e}$$

As discussed before, since it is more difficult to satisfy latency compared to bandwidth, we assign a higher weight to latency. Two communicating nodes that have low latency and high bandwidth are close to each other in terms of the distance metric, and are placed in the same cluster.

Once the clusters have been formed, for every communication trace that is cut across a cluster boundary, two dummy nodes are added to the respective clusters. If two edges share either a source or sink node, then only two dummy nodes are introduced instead of four. For example, in Figure 16 the top two edges that are cut share a common source in the left hand side cluster. Hence, only two dummy nodes (that are labelled as "A" in the figure) are introduced. The latency constraint on the original communication trace is split in half across the edges attached to the pair of dummy nodes. The bandwidth constraint is duplicated on the edges. The MILP formulation for topology design is then utilized to generate the partial solution for each cluster. In the figure, we assume that the routers have four ports, and are on the four sides of the rectangle. The full topology is generated from the partial solution by adding physical links between ports of routers that are in neighboring clusters, and are attached to identically named dummy nodes. For example, in Figure 16, routers R1 and R3 that are in different clusters are attached together with physical link since they both have a dummy node named "A" assigned to a port.

V. Results

In this section, we present the results obtained by execution of our techniques on multimedia benchmark applications. In Section V-A, we discuss the benchmark applications, in Section V-B, we discuss the experimental setup, and in Section V-C we present and discuss the results.

A. Benchmark applications

We generated custom NoC architectures for four multimedia benchmarks namely, mp3 audio encoder, mp3 au-

Graph	Graph ID	Nodes	Edges
mp3 decoder	G1	5	3
263 encoder	G2	7	8
mp3 encoder	G3	8	9
263 decoder	G4	9	8
263 enc mp3 dec	G5	12	12
mp3 enc mp3 dec	G6	13	12
263 dec mp3 dec	G7	14	16
263 enc mp3 enc	G8	15	17
263 enc 263 dec	G9	16	16
263 dec mp3 enc	G10	17	17

TABLE I
GRAPH CHARACTERISTICS

Node	263 dec mp3 dec	263 enc mp3 dec	mp3 enc mp3 dec
0	VLD	ME	FP
1	IQ	DCT	FFT
2	IDCT	FP	FILTER
3	MC	IDCT	MDCT
4	ADD	MC	ITER. ENC.1
5	MEM 1	VLE	ITER. ENC.2
6	MEM 2	MEM	BIT RES 1
7	HUFF 1	BIT RES 1	BIT RES 2
8	HUFF 2	BIT RES 2	BIT RES 3
9	BIT RES 1	IMDCT	BIT RES 4
10	BIT RES 2	SUM	IMDCT
11	IMDCT	BUF	SUM
12	SUM		BUF
13	BUF		

TABLE II
NODE DESCRIPTIONS

No.	Graph	Power (μW)			Routers		
		Cluster	Mesh	Ratio	Cluster	Mesh	Ratio
1	G1	2.622	7.363	2.80	1	5	5
2	G2	108.3	291.4	2.69	2	7	3.5
3	G3	5.7	10.51	1.84	2	8	4
4	G4	5.722	12.51	2.18	3	9	3
5	G5	110.4	273.7	2.47	4	12	3
6	G6	8.157	18.02	2.21	5	13	2.6
7	G7	8.535	22.27	2.60	5	14	2.8
8	G8	155.2	277.0	1.78	5	15	3
9	G9	115.6	296.7	2.56	5	16	3.2
10	G10	11.54	28.63	2.15	6	17	2.8

TABLE V
COMPARISON OF CLUSTERING, AND MESH

No.	Graph	Power (μW)			Routers		
		Cluster	QNoC	Ratio	Cluster	QNoC	Ratio
1	G1	2.622	2.627	1.00	1	2	2
2	G2	108.3	216.1	1.99	2	4	2
3	G3	5.7	6.368	1.11	2	4	2
4	G4	5.722	6.453	1.12	3	3	1
5	G5	110.4	244.9	2.21	4	6	1.5
6	G6	8.157	12.35	1.51	5	5	1
7	G7	8.535	22.37	2.62	5	6	1.2
8	G8	155.2	155.7	1.00	5	5	1
9	G9	115.6	352.4	3.04	5	7	1.4
10	G10	11.54	25.86	1.95	6	7	1.1

TABLE VI
COMPARISON OF CLUSTERING, AND QNOC

dio decoder, H.263 video encoder, and H.263 video decoder algorithms. In addition, we obtained results for six other benchmarks by mapping combinations of two applications from the above mentioned benchmarks simultaneously. The benchmarks are shown in Table I. The communication trace graphs for the benchmarks were obtained from the work presented by Hu et al. [49].

B. Experimental setup

In our experimental setup, we obtained results for router architectures with 5 and 4 ports, respectively. As discussed in Section I-A, the power consumption in 100 nm technology, for the input and output port was estimated to be $328nW/Mbps$ and $65.5nW/Mbps$, respectively. The link power consumption was estimated to be $79.6nW/Mbps/mm$. We utilized the Xpress-MP optimizer [57] to solve the MILP problems. The solver was configured with a timeout of 8 hours for the floorplanning stage, and a timeout of 12 hours for the NoC architecture generation stage. If the solver failed to find the optimal solution, the best available solution generated within the timeout criterion was accepted. We obtained best results when:

- the minimum distance between two routers was set to one half the length of the maximum sized node,
- the distance of a node from the router to which it can be mapped was set to the length of the maximum sized node,
- the sizes of the clusters were limited to 9 nodes for the clustering based heuristic.

All results were obtained on a 950 MHz SPARC processor.

Graph	Area ratio (Mesh over Custom)	Runtime (sec)	
		Custom	Mesh
G1	1.26	< 1	< 1
G2	1.09	2	13
G3	1.21	17	56
G4	1.45	9	31
G5	1.74	507	9987
G6	1.56	13376	28800(t.o)
G7	1.39	2383	13746
G8	1.44	28800(t.o)	28800(t.o)
G9	1.36	28800(t.o)	28800(t.o)
G10	1.38	28800(t.o)	28800(t.o)

TABLE III
RESULTS FOR FLOORPLANNING

C. Results and discussion

In this section, we present and discuss the results for the floorplanning, and the topology generation and routing stages.

C.1 Floorplanning stage

Figures 17, 22, and 27 present the communication trace graphs for 263 dec-mp3 dec, 263 enc-mp3 dec, and mp3 enc-mp3 dec benchmarks, respectively. The edges of the graphs are annotated with bandwidth requirement in $Kbps$. The node descriptions are depicted in Table II. Figures 18, 23, and 28 present the corresponding floorplans obtained by executing our MILP based floorplanner on the benchmarks, for custom architectures. Figures 19, 24, and 29 present the corresponding floorplans for mesh architectures. The floorplanner places highly communicating nodes close to each other. For example, in Figure 18, nodes 1 and 2 that have high communication bandwidth, are placed next to each other.

Table III presents the ratio of the area consumption of mesh based topologies over that of custom topologies, and the runtimes of the floorplanning stage for custom and mesh topologies. On an average, the mesh topology consumed 1.38 times the area consumed by custom topology.

No.	No. Ports	Graph	No of Clusters	Power (μW)			Routers			Runtime (secs)	
				MILP	CLUSTER	Ratio	MILP	CLUSTER	Ratio	MILP	CLUSTER
1	5	G1	1	2.622	2.622	1	1	1	1	< 1	< 1
2	5	G2	1	108.3	108.3	1	2	2	1	330	330
3	5	G3	1	5.7	5.7	1	2	2	1	1720	1720
4	5	G4	1	5.722	5.722	1	3	3	1	2318	2318
5	5	G5	2	179.5	110.9	0.61	5	4	0.8	41200 (t.o)	1103
6	5	G6	2	8.635	8.157	0.94	5	5	1	41200 (t.o)	2099
7	5	G7	2	11.91	8.535	0.71	5	5	1	41200 (t.o)	35522
8	5	G8	2	170.7	155.2	0.90	5	5	1	41200 (t.o)	1559
9	5	G9	2	245.4	115.6	0.47	7	5	0.7	41200 (t.o)	35467
10	5	G10	2	15.52	11.54	0.74	7	6	0.8	41200 (t.o)	1540
11	4	G1	1	2.631	2.631	1	2	2	1	< 1	< 1
12	4	G2	1	138.0	138.0	1	4	4	1	347	347
13	4	G3	1	5.944	5.944	1	3	3	1	1206	1206
14	4	G4	1	5.722	5.722	1	4	4	1	22222	22222
15	4	G5	2	194.6	140.7	0.72	5	5	1	41200(t.o)	2502
16	4	G6	2	10.94	12.47	1.12	6	6	1	41200(t.o)	41200(t.o)
17	4	G7	2	13.90	8.664	0.62	6	6	1	41200(t.o)	36733
18	4	G8	2	158.6	209.4	1.31	7	7	1	41200(t.o)	41200(t.o)
19	4	G9	2	241.3	147.2	0.61	7	7	1	41200(t.o)	41200(t.o)
20	4	G10	2	17.22	11.97	0.69	8	8	1	41200(t.o)	36544

TABLE IV
COMPARISON OF MILP, AND CLUSTERING

No.	No. Ports	Graph	Lower bound (μW)		Cluster (μW)	Ratio	
			MILP (ML)	Cluster (CL)	(C)	(C/ML)	(C/CL)
1	5	G1	2.622	2.622	2.622	1	1
2	5	G2	108.3	108.3	108.3	1	1
3	5	G3	5.7	5.7	5.7	1	1
4	5	G4	5.722	5.722	5.722	1	1
5	5	G5	93.26	110.9	110.9	1.18	1
6	5	G6	5.50	8.15	8.15	1.48	1
7	5	G7	8.10	8.53	8.535	1.05	1
8	5	G8	116.9	155.2	155.2	1.32	1
9	5	G9	90.74	115.6	115.6	1.27	1
10	5	G10	10.97	11.54	11.54	1.05	1
11	4	G1	2.631	2.631	2.631	1	1
12	4	G2	138.0	138.0	138.0	1	1
13	4	G3	5.944	5.944	5.944	1	1
14	4	G4	5.722	5.722	5.722	1	1
15	4	G5	121.4	140.7	140.7	1.15	1
16	4	G6	8.20	10.4	12.47	1.52	1.19
17	4	G7	6.88	8.66	8.664	1.25	1
18	4	G8	102.1	134.3	209.4	2.05	1.55
19	4	G9	80.33	113.7	147.2	1.83	1.29
20	4	G10	9.01	11.97	11.97	1.32	1

TABLE VII
COMPARISON OF CLUSTERING FINAL SOLUTION WITH MILP AND CLUSTERING LOWER BOUNDS

Since the cores of the mesh are aligned in a grid, mesh topologies occupy more area compared to custom topologies. In the table, "t.o" denotes that the solver did not converge to an optimal solution within the timeout period of 8 hours. The longer running time for mesh may be attributed to the extra constraints required in the formulation to generate mesh topologies.

C.2 Topology generation and routing stage

Table IV compares the results obtained for the MILP and clustering based techniques, respectively. In the table, column 1 gives the serial number, column 2 denotes the given router architecture, column 3 specifies the benchmark application, column 4 denotes the number of clusters for each benchmark, columns 5 and 6 present the total power consumption of solutions produced by MILP formulation, and the clustering technique respectively, column 7 gives the ratio of power consumption of the clustering technique solutions over the MILP solutions, columns 8 and 9 present the router requirements of MILP and clustering techniques, respectively, column 10 denotes the ratio of routers required by the clustering solutions over the MILP solutions, and finally columns 11 and 12 denote the run times of the MILP and clustering techniques, respectively. In the table, "t.o" denotes that the solver did not converge to an optimal solution within the timeout period of 12 hours.

The clustering technique performed better than MILP for the timeout criterion of 12 hours. Due to its smaller size compared to the benchmark application, the solver was able to generate optimal solutions for the clusters. On the other hand, the solver timed out for many benchmarks when only the MILP was invoked. On an average, the clustering technique produced results that consumed only 85%, and 96% of the power and number of routers, respectively, in comparison to MILP.

The custom topologies of the three benchmarks (263 decmp3 dec, 263 enc-mp3 dec, and mp3 enc-mp3 dec benchmarks) produced by clustering based techniques are shown in Figures 20, 25, and 30, respectively, for 5 port router architectures, and Figures 21, 26, and 31, respectively, for 4 port router architectures.

We also compared our approach of synthesizing customized NoC designs for application specific SoC architectures against solutions with mesh based NoC topologies. The floorplans for custom and mesh architectures were obtained by invoking the MILP solver on the formulation presented in Section III-A. Once the floorplan was obtained, we obtained the power and router consumption of a regular mesh topology by considering the shortest distance in terms of the number of routers from the source node to the sink node for each trace. The value thus calculated serves as a lower bound on the power consumption of the regular mesh topology. Similarly, we also obtained the number of routers and power consumption for the QNoC architecture [44]. The QNoC architecture is identical to a mesh topology except that it permits multiple cores to be attached to routers that are on the periphery of the mesh. Tables V and VI show the results of the comparative study for each

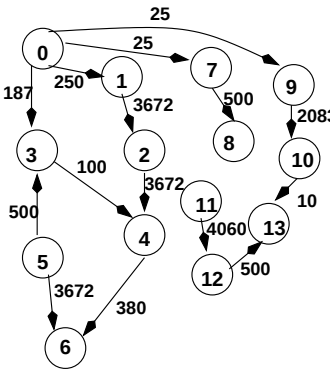


Fig. 17. 263 dec mp3 dec: Communication Trace Graph

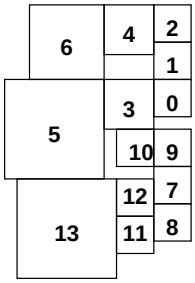


Fig. 18. Custom layout for 263 dec mp3 dec

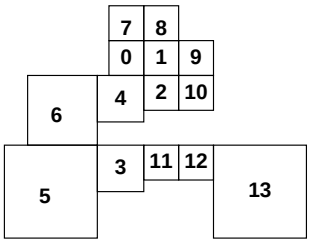


Fig. 19. Mesh layout for 263 dec mp3 dec

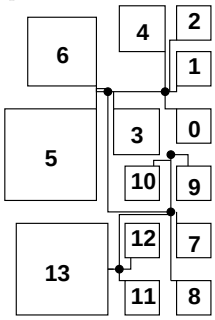


Fig. 20. 5 port topology for 263 dec mp3 dec

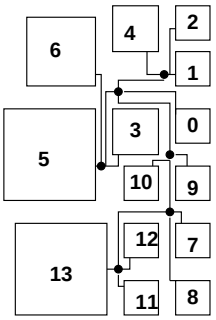


Fig. 21. 4 port topology for 263 dec mp3 dec

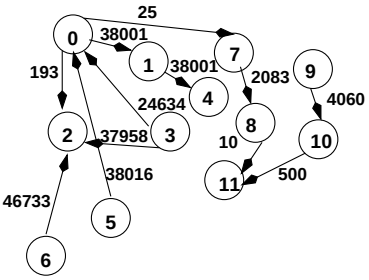


Fig. 22. 263 enc mp3 dec:Communication Trace Graph

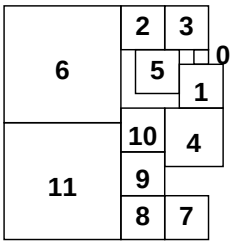


Fig. 23. Custom layout for 263 enc mp3 dec

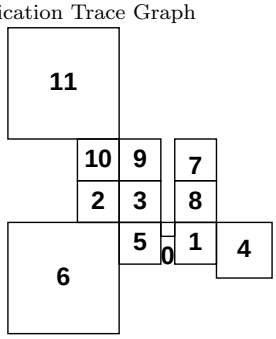


Fig. 24. Mesh layout for 263 enc mp3 dec

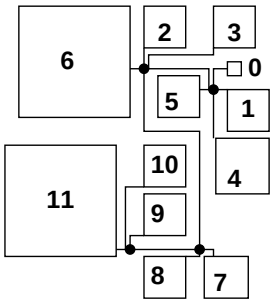


Fig. 25. 5 port topology for 263 enc mp3 dec

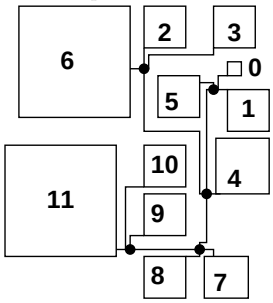


Fig. 26. 4 port topology for 263 enc mp3 dec

benchmark application. The number of ports in the routers was 5 in both cases. The number of nodes in the communication trace graph place a lower bound on the number of routers in both regular mesh and QNoC based topologies. The regular mesh based topology on an average consumes over 2.3 times more power and requires 3.5 times as many routers as a customized topology. The QNoC based topology on an average consumes 1.75 times more power and requires 1.4 times as many routers as a customized topology. The pre-designed physical connections in both the mesh based topologies force the communication traces to pass through more routers, thus, leading to the increased power consumption.

We compared the final solution generated by the clustering based heuristic with the theoretical lower bound on the power consumption of the MILP and clustering based formulations, respectively (see Table VII). We would like to emphasize that for every graph for which the MILP formulation resulted in a time out (rows 5-10 and 15-20), the lower bound is strictly a theoretical value, and it denotes the best lower bound that was generated by the formulation for a particular graph within the specified time. The lower bound for the clustering based technique for a graph is the summation of the lower bounds for the individual formulations for different clusters of the same graph. On an average for the larger graphs (rows 5-10 and 15-20) the final result of the clustering based technique is 1.37 (standard deviation of 0.31) and 1.08 (standard deviation of 0.17) times of the lower bounds due to the MILP and clustering based formulations, respectively. Thus, the clustering based heuristic generates results that are within 40 % of the theoretical optimal lower bound.

We also evaluated the number of additional virtual channels that are required for the custom architectures synthesized by our techniques. We found that for our set of applications, deadlock did not occur in any of the synthesized designs. We next measured the maximum number of traces flowing through any port of a router in the topology. This value acts as an upper bound on the number of virtual channels required in the design if a deadlock were to occur. We found that for most of the cases the number of traces were either 1 or 2. In only 5 (out of a total of 20) of the

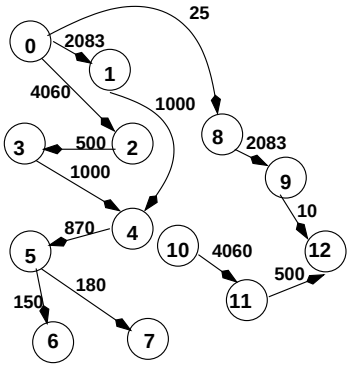


Fig. 27. mp3 enc mp3 dec:Communication Trace Graph

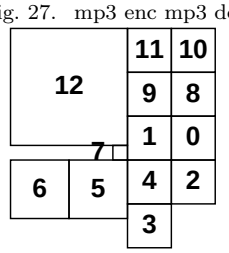


Fig. 28. Custom layout for mp3 enc mp3 dec

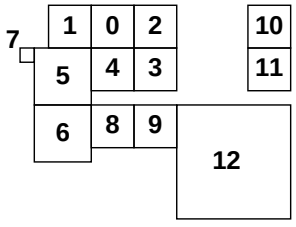


Fig. 29. Mesh layout for mp3 enc mp3 dec

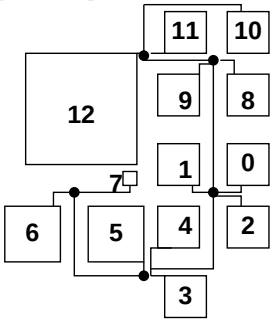


Fig. 30. 5 port topology for mp3 enc mp3 dec

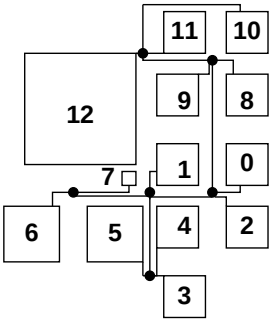


Fig. 31. 4 port topology for mp3 enc mp3 dec

cases we had some ports that supported 3 traffic traces. Warnakulasuriya et al. [58] show that the deadlock probability in irregular networks becomes negligible with three virtual channels. Therefore, we can conclude that for the multimedia benchmarks utilized in this paper the number of additional virtual channels required after synthesis are minimal.

VI. Discussion

Our techniques synthesize an application specific NoC with an objective of minimizing the communication power subject to the performance constraints. The techniques account for both the physical link and router power consumption. The techniques incorporate a system-level floorplanning stage to calculate the link power accurately. As demonstrated in the results section, the custom topologies generated by our techniques perform better than both traditional mesh based and QNOC based NoC architectures in terms of power and resource consumption. In the comparative study we focussed on the power and resource requirement due to the interconnection architecture. The overall advantage will be smaller if the power and area consumption of the computation architecture is also included in the comparison.

The discussion on router characterization (Section I-A) assumed a particular router architecture with data width of 32, 4 virtual channels at every port (2 for input and 2 for output), depth of virtual channels at 8, a packet size of 256 bits, and that performed wormhole switching. However, our techniques are not limited to the particular router architecture. They are applicable for any router architecture in which the input and output port power varies linearly with bandwidth. We expect that these characteristics are applicable to all other router architectures, and therefore so are our techniques.

The discussion on the techniques and experimental results assumed that the NoC is designed with homogeneous router architectures. However, the techniques can easily be extended to heterogeneous router architectures. For example, if the maximum number of ports in a router that is available in the IP library are 8, the formulation for topology generation can initially assume that all routers have 8 ports. Once the topology has been generated, additional un-utilized ports can be eliminated from all the routers.

The technique requires that the router architecture be characterized for its power consumption. During characterization the router architecture has a certain number of virtual channels (two in our case). The final number of virtual channels might vary due to the possibility of deadlocks. However, for the multimedia benchmarks utilized in this paper, our techniques did not result in deadlocks for any design. Further, only 25 % of the designs had routers with more than 3 traces routed through a single port. This number acts as an upper bound on the required number of virtual channels to avoid a deadlock should it occur. Further, even for generic NoC architectures increasing the number of virtual channels results in diminishing performance improvements. Therefore, the router architecture can be characterized with a fixed number of virtual channels, and the deviation from the architecture is expected to be minimal.

Although our objective functions are aimed at dynamic power minimization, we do address router utilization. We reduce the number of available routers at floorplanning stage by eliminating redundant routers. At the interconnection architecture design stage we reduce the number of hops for each traffic trace and thereby utilizing minimum number of routers.

Our techniques are aimed at application specific SoC architectures where the computation elements demonstrate well defined inter-core communication characteristics. Consequently, the communication requirements can be specified by a directed communication graph, and an optimized NoC topology with static routes can be designed by utilizing our techniques. Our techniques do not currently address fault tolerant topologies. Automated design techniques for fault tolerant topologies that support static routing would be focus of future work. As we generate custom topologies, adaptive routing techniques that can

be readily applied to regular topologies cannot be easily integrated into our architectures.

The NoC is designed after the computation architecture has been finalized. Consequently, even before the NoC is generated the bandwidth production and consumption requirements of the various computation architecture cores are known. Therefore, we assume that cores can produce and consume the required bandwidth. The only requirement that we place on the NoC is that the unit router input and output port should be able to consume and produce the required core bandwidth, respectively. In the case that the computation architecture has a wide variation in the bandwidth requirements, the designer can choose to develop a NoC architecture with either average or worst case communication traffic by specifying respective communication trace graphs.

We have compared against known mesh based topologies (regular and QNoC) that have been utilized by others. Regular mesh based topologies require the same number of routers as there are cores. QNoC permits only routers that are on the boundary of a mesh to be connected to multiple cores. In the custom topologies generated by our techniques multiple cores can be connected to routers. As the power consumption of the unit routers are several times that of the physical links, custom topologies optimize both the power consumption and the latency of the NoC. This advantage would diminish as the proportion of power consumption of the physical links to the power consumption of the routers increases.

The MILP based techniques do not generate an optimal result in 12 hours for larger sized graphs. However, the clustering based heuristic are able to generate results is almost all cases (3 exceptions out of 20) in a reasonable amount of time. Further, the overall results produced by the clustering based technique were better than the MILP results both in terms of power consumption (96%) and router requirements (85%). Thus, the clustering based technique is an effective heuristic for generating NoC designs for larger SoC architectures with hundreds of nodes. In our experiments we utilized a cluster size of 9 nodes. The run time of the clustering based technique can be improved by reducing the cluster size (to eight or seven nodes). Thus, for a SoC with 100 components, our heuristic technique would operate on about 12 clusters (of size 8) to generate the overall NoC architecture.

Technology scaling will result in an increase in the static power consumption due to the routers and dynamic power consumption due to the physical links. Our techniques account for the contribution of physical links toward the total power consumption. Static power consumption can be minimized by two techniques. First, router ports that do not support any traffic traces can be eliminated from the design. Thus, the final topology would consist of heterogeneous router architectures. Virtual channels are chief contributors of leakage power in a router architecture [34]. Their contribution can be reduced by over 80 % by deploying run time power management schemes such as those proposed by Chen et al. [34].

The formulation for floorplanning presented in Section III-A does not address aspect ratios and orientations of individual cores. At the floorplanning stage the contribution of the paper is in terms of a unique cost function that addresses NoC specific constraints, and extensions for addressing mesh based layouts. Please refer to [52] [53] [54] for more complete floorplanning formulations. All of these formulations can be combined with the cost function proposed in the paper to generate NoC specific layouts.

In nanoscale technologies, architectures are expected to be inherently faulty. The traditional techniques of introducing redundancy to improve fault tolerance will prove expensive in terms of leakage power as well as area. Novel techniques for design of fault tolerant NoC is an open problem.

Our paper addresses the NoC design in isolation. Integration of computation architecture design with NoC synthesis has the potential for larger power savings.

VII. Conclusion

In this paper, we defined the application specific NoC synthesis problem and proposed linear programming based solutions. We addressed the complexity of NoC synthesis problem by dividing it into two stages namely, floorplanning and interconnection network generation. We presented optimal MILP formulations for the two stages, and presented a clustering based heuristic for the second stage to reduce the run time of the formulation. We performed extensive experimentation to validate the quality of our techniques. The optimal MILP formulation timed out for many benchmarks. On the other hand, our clustering based technique was able to generate results with superior quality in reasonable time. On an average, the clustering technique consumed only 85% of the power and 96% of the router resources respectively, compared to the corresponding results generated by the MILP formulation. We also compared the custom topologies synthesized by our technique with regular mesh and QNOC based interconnection networks. The mesh and QNoC based topologies on an average consumed 2.3 and 1.75 times more power, and required over 3.5 and 1.4 times the router resources as compared to our custom topologies, respectively.

References

[1] "International Technical Roadmap for Semiconductors". Technical report, http://public.itrs.net/, 2004.

[2] D. Sylvester and K. Keutzer. "A Global Wiring Paradign for Deep Submicron Design". *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, pages 242–252, February 2000.

[3] R.Ho, K. Mai and M. Horowitz. "The Future of Wires". *Proceedings of IEEE*, pages 490–504, April 2001.

[4] William J. Dally and Brian Towles. "Route Packet, Not Wires: On-Chip Interconnection Networks". In *Proceedings of DAC*, June 2002.

[5] Luca Benini and Giovanni De Micheli. "Networks on Chips: A New SoC Paradigm". *IEEE Computer*, pages 70–78, January 2002.

[6] M.B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J-W Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen M. Frank, S. Amarasinghe, and A. Agrawal. "The RAW Microprocessor: A Computational Fabric for Software Circuits and General-

Purpose Programs". *IEEE Micro*, pages 25–35, March-April 2002.

[7] Antoine Jalabert, Srinivasan Murali, Luca Benini, and Giovanni De Micheli. "xpipesCompiler: A tool for instantiating application specific Networks on Chip". In *DATE*, 2004.

[8] S. Prakash and A.C. Parker. "SOS: Synthesis of Application-Specific Heterogeneous Multiprocessor Systems". *Journal of Parallel and Distributed Computing*, 16:338–351, 1992.

[9] B.P. Dave, G. Lakshimnarayana and N.K. Jha. "COSYN: Hardware-Software Cosynthesis for Heterogeneous Distributed Embedded Systems". *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, 7(1), March 1999.

[10] R.P. Dick and N.K. Jha. "MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Cosynthesis of Distributed Embedded Systems". *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 17(10), October 1998.

[11] N. Banerjee, P. Vellanki, and K. S. Chatha . "A Power and Performance Model for Network-on-Chip Architectures ". In *Proceedings of DATE*, Paris, France, February 2004.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[13] R. Ravi, M. V. Marathe, S.S. Ravi, D. J. Rosenkrantz, H. B. Hunt III . "Approximation Algorithms for Degree-Constrained Minimum-Cost Network-Design Problems" . *Algorithmica*, 31(1):58–78, 2001.

[14] W.J. Dally and C.L. Seitz. "Deadlock-free message routing in multiprocessor interconnection networks". *"IEEE Transactions on Computers"*, C-36(5):547–553, 1987.

[15] K. Srinivasan, K.S. Chatha and G. Konjevod. "Linear Programming based Techniques for Synthesis of Network-on-Chip Architectures" In *Proceedings of International Conference on Computer Design (ICCD)*, San Jose, CA, 2004.

[16] K. Srinivasan, and K.S. Chatha. "A Methodology of Layout Aware Design and Optimization of Custom Network-on-Chip Architectures" In *Proceedings of International Symposium on Quality Electronic Design (ISQED)*, San Jose, CA, 2006.

[17] P.Guerrier and A.Greiner. "A Generic Architecture for On-Chip Packet-Switched Interconnections". In *DATE*, Paris, France, March 2000.

[18] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, and M. Millberg. "Network on Chip: An architecture for billion transistor era". In *Proceedings of IEEE NorChip Conference*, November 2000.

[19] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabeay and A. Sangiovanni-Vincentelli. "Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design". In *Proceedings of Design Automation Conference*, pages 667–672, June 2001.

[20] Shashi Kumar, Axel Jantsch, Mikael Millberg, Johny Oberg, Juha-Pekka Soininen, Martti Forsell, Kari Tiensyrj Ahmed Hemani. "A Network on Chip Architecture and Design Methodology". In *IEEE Computer Society Annual Symposium on VLSI*, Pittsburg, Pennsylvania, April 2002.

[21] A. Andriahantenaina and A. Greiner. "Micro-network for SoC: Implementation of a 32-port SPIN network". In *DATE*, Munich, Germany, March 2003.

[22] A. Andriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino. "SPIN: a Scalable, Packet Switched, On-Chip Micro-network". In *DATE*, Munich, Germany, March 2003.

[23] D. Siguenza-Tortosa and J. Nurmi. "Proteo: A New Approach to Network-on-Chip". In *Proceedings of IASTED International Conference on Communication Systems and Network*, Malaga, Spain, 2002.

[24] Siguenza-Tortosa, D. and Nurmi, J. "VHDL-based simulation environment for Proteo NoC". In *High-Level Design Validation and Test Workshop*, Paris, France, October 2002.

[25] M. Dall'Osso, G. Biccari, L. Giovanninni, D. Bertozzi, and L. Benini. "xpipes: a Latency Insensitive Parameterized Network-on-Chip Architecture for Multi-Processor SoCs". In *Proceedings of ICCD*, San Jose, CA, October 2003.

[26] Mikael Millberg, Erland Nilsson, Rikard Thid, Shashi Kumar, and Axel Jantsch. "The Nostrum backbone - a communication protocol stack for networks on chip". In *VLSI Design Conference*, Mumbai, India, January 2004.

[27] Mikael Millberg, Erland Nilsson, Rikard Thid, and Axel Jantsch. "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip.". In *DATE*, pages 890–895, February 2004.

[28] John Dielissen, Andrei Rădulescu, Kees Goossens, and Edwin Rijpkema. Concepts and implementation of the Philips network-on-chip. In *IP-Based SOC Design*, November 2003.

[29] E. Rijpkema, K. G. W. Goossens, and A. Radulescu. "Trade Offs in the Design of a Router with Both Guaranteed Best-Effort Services for Networks on chip". In *DATE*, 2004.

[30] N. Banerjee P. Vellanki and K. S. Chatha. Quality-of-service and error control techniques for mesh based network-on-chip architectures. *Integration, the VLSI Journal*, 38:353–382, 2005.

[31] D. Bertozzi, L. Benini, and G. De Micheli. "Low power error resilient encoding for on-chip data buses". In *DATE*, 2003.

[32] H. Zimmer and A. Jantsch. "A Fault Model Notation and Error-Control Scheme for switch-to-Switch Buses in a Network-on-Chip". In *ISSS/CODES*, 2003.

[33] F. Worm, P. Ienne, P. Thiran, G. De Micheli. "An Adaptive Low-Power Transmission Scheme for On-Chip Networks". In *Proceedings of ISSS*, Kyoto, Japan, 2002.

[34] X. Chen, and L-S Peh. "Leakage Power Modeling and Optimization in Interconnection Networks". In *Proceedings of ISLPED*, Seoul, Korea, 2003.

[35] T. Simunic and S. Boyd. "Managing Power Consumption in Networks on Chips". In *Proceedings of DATE*, Paris, France, 2002.

[36] E. Nilsson and J. Oberg. "Reducing Power and Latency in 2-D Mesh NoC using Globally Pseudochronous and Locally Synchronous Clocking". In *Proceedings of ISSS-CODES*, 2004.

[37] J. Duato, S. Yalamanchili, L. Ni . *"Interconnection Networks, an Engineering Approach"*. IEEE Computer Society, 1997.

[38] H.J. Seigel. "A model of SIMD machines and a comparison of various interconnection networks". *"IEEE Transactions on Computers"*, 28(12):907–917, December 1979.

[39] W.J. Dally. "Performance analysis of k-ary n-cube interconnection netwrok". *"IEEE Transactions on Computers"*, 39(6):775–785, June 1990.

[40] J.F. Draper and J. Ghosh. "A Comprehensive Analytical Model for Wormhole Routing in Multicomputer Systems". *"Journal of Parallel and Distributed Computing"*, 23:202–214, 1994.

[41] A.G. Wassal and M.A. Hasan. "Low-power system-level design of VLSI packet switching fabrics". *IEEE Transactions on CAD*, 20:723–738, June 2001.

[42] Terry T. Ye, Luca Benini and Giovanni De Micheli. "Analysis of Power Consumption on Switch Fabrics in Network Routers". In *Proceedings of DAC*, 2002.

[43] H-S Wang, L-S Peh and S. Malik. "Orion: A Power-Performance Simulator for Interconnection Network". In *International Symposium on Microarchitecture*, Istanbul, Turkey, November 2002.

[44] Evgeny Bolotin, Israel Cidon, Ran Ginosar and Avinoam Kolodny. "Cost considerations in Network on Chip". November 2003.

[45] D. Pamunuwa, J. Oberg, L. R. Zheng, M. Millberg, A. Jantsch, and H. Tenhunen. "Layout, performance and power trade-offs in mesh-based network-on-chip architectures". In *IFIP International Conference on Very Large Scale Integration(VLSI-SOC), Darmstadt, Germany*, pages 362–367, December 2003.

[46] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. "Efficient Synthesis of Networks On Chip". In *ICCD*, 2003.

[47] T. Lei and S. Kumar. "A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture". In *Proceedings of Euromicro Symposium in Digital Systems Design (DSD)*, 2003.

[48] G. Ascia, V. Catania, and M. Palesi. "Multi-objective Mapping for Mesh-based NoC Architectures". In *Proceedings of ISSS-CODES*, 2004.

[49] J. Hu and R. Marculescu. "Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints". In *ASP-DAC*, 2003.

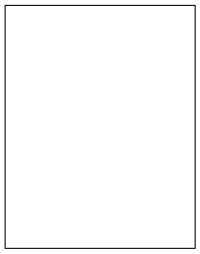
[50] Jingcau Hu, and Radu Marculescu. "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures". In *DATE*, 2003.

[51] Jingcau Hu, and Radu Marculescu. "Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints". In *DATE*, 2004.

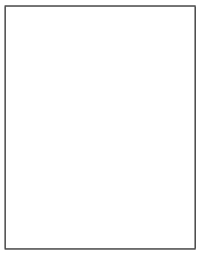
[52] S.M Sait and H. Youssef. *VLSI Physical Design Automation: Theory and Practice*. McGraw-Hill Inc, 1994.

[53] P. Chen, and E.S Kuh. "Floorplan Sizing by Linear Program-

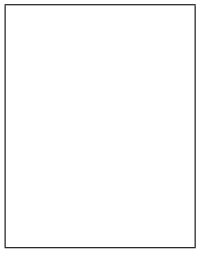
ming Approximation". In *Proceeding of DAC*, Los Angeles, California, June 2000.
[54] J.G Kim, and Y.D Kim. "A Linear Programming Based Algorithm for Floorplanning in VLSI Design". *IEEE Transactions on CAD*, 22(5), 2003.
[55] D'andrade R. "U-Statistic Hierarchical Clustering". *Psychometrica*, 4(58-67), 1978.
[56] . http://www.analytictech.com/networks/hiclus.htm . 2004.
[57] . "www.dashoptimization.com" . 2004.
[58] Sugath Warnakulasuriya, and Timothy Mark Pinkston. "Characterization of Deadlocks in Irregular Networks". *Journal of Parallel and Distributed Systems*, 62(1):61–84, 2002.



Krishnan Srinivasan is a PhD candidate in the Department of Computer Science and Engineering, at Arizona State University. He obtained his Masters degree in Electrical Engineering from the same university in 2002, and a Bachelors degree in Electrical Engineering from the Regional Institute of Technology, India, in 1999. Srinivasan's research interests are in the field of power and performance optimization of System-on-Chip(SoC) and Network-on-Chip (NoC) Architectures. His current research focuses on mathematical models and approximation algorithms for low power SoC and NoC design. Srinivasan is a student member of the IEEE and ACM.



Karam S. Chatha is an Assistant Professor in the Department of Computer Science and Engineering at the Arizona State University. His research interests are in all aspects of application specific digital system design, including architectures, design methodologies, and computer-aided design (CAD) tools. In particular, he has focused on Network-on-Chip (NoC) design, multiprocessor System-on-Chip (MPSoC) design, hardware-software co-design, and reconfigurable and adaptive computing. Karam received the "Best Paper Award" at International Workshop on Field Programmable Logic (FPL) 1999. He received his B.E. (Hons) degree in Computer Technology from Bombay University in 1993, M.S. and Ph.D. in Computer Science and Engineering from University of Cincinnati in 1997 and 2001, respectively. Karam is a member of the ACM and IEEE. Email: kchatha@asu.edu



Goran Konjevod is an Assistant Professor in the Department of Computer Science and Engineering at the Arizona State University. His research interests include various areas of theoretical computer science, discrete mathematics and operations research. Recently, the main focal point of his work has been theory and applications of linear programming relaxations for combinatorial optimization problems. Goran Konjevod presented (jointly with R. D. Carr of Sandia National Laboratories) a tutorial on Polyhedral Combinatorics at the 2004 INFORMS meeting in Denver. He received his B.S. in Mathematics from University of Zagreb in 1995, and M.S. in Algorithms, Combinatorics and Optimization (in 1998) and Ph.D. in Applied Mathematics (in 2000) from Carnegie Mellon University. Goran Konjevod is a member of the SIAM and MAA. Email: goran@asu.edu