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

Krishnan Srinivasan, *Student Member, IEEE*, Karam S. Chatha, *Member, IEEE*, and Goran Konjevod

***Abstract*—Application-specific system-on-chip (SoC) design offers the opportunity for incorporating custom network-on-chip (NoC) architectures that are more suitable for a particular application, and do not necessarily conform to regular topologies. This paper presents novel mixed integer linear programming (MILP) formulations 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 power consumption of the NoC architecture is determined by both the physical links and routers. The power consumption of a physical link is dependent upon the length of the link, which 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 also present a clustering-based heuristic technique for the second stage to reduce the run times of the MILP formulation. We analyze the quality of the results and solution times of the proposed techniques by extensive experimentation with realistic benchmarks and comparisons with regular mesh-based NoC architectures.**

***Index Terms*—Design automation, integrated circuit interconnection, multiprocessor interconnection.**

I. Introduction

INTERNATIONAL Technological Roadmap for Semiconductors [1] predicts that in future high-end system-on-chip (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 resistance–inductance–capacitance (*RLC*) effects. Network-on-chip (NoC) has been proposed as a solution for the SoC global communication challenges in nanoscale technologies [4], [5]. NoC supports asynchronous transfer of packets. It can support high-communication bandwidth by distributing the propagation delay across multiple switches, thus pipelining the signal transmission. The lower left-hand-side half of Fig. 1 shows a SoC architecture. In the figure, the "P/M" blocks denote digital signal processing (DSP), application-specific integrated circuit (ASIC) or storage cores (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.

NoC architectures can be designed with both regular and custom (or irregular) topologies. The primary advantage of a regular NoC architecture is topology reuse and reduced design time. They are suitable for general purpose architectures, such as the RAW processor [6] that include homogeneous cores. This paper addresses the design of a NoC in the context of application-specific SoC architectures. Regular topologies assume that every core has equal communication bandwidth with every other core which does not hold in custom SoCs. Application-specific SoC architectures consist of heterogeneous cores and memory elements which have vastly different sizes. Consequently, 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 a large amount of area overhead.

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. For such architectures, as the results of this 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. In custom topologies, the router architecture itself is regular and can be easily parameterized for reuse (on number of ports, width of physical links, number of virtual channels, and so on).

The complexity of automated system-level design can be addressed by decomposing it into two stages: 1) computation architecture design and 2) physical design and NoC architecture synthesis. The output of the computation architecture design stage is a collection of processing and memory cores and a mapping of the computation and storage operations on the respective cores. This problem has been widely addressed in the system-level synthesis [8] community. The paper focuses on the problem of automated application-specific NoC design. We also extend our techniques to mesh-based interconnection architectures and compare them against custom topologies.

NoC design flow is shown in Fig. 1. The input to the NoC 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 labelled as "P/M" blocks. Each "P/M" block is uniquely identified by a node number "n i." The physical dimensions of the blocks are also specified. 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

Manuscript received August 22, 2004; revised December 15, 2004 and July 29, 2005. This work was supported in part by the National Science Foundation under CAREER Award CCF-0546462, IIS-0308268 and in part by Consortium for Embedded Systems.

The authors are with the Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: ksrinivasan@asu.edu; kchatha@asu.edu; goran@asu.edu).

Digital Object Identifier 10.1109/TVLSI.2006.871762

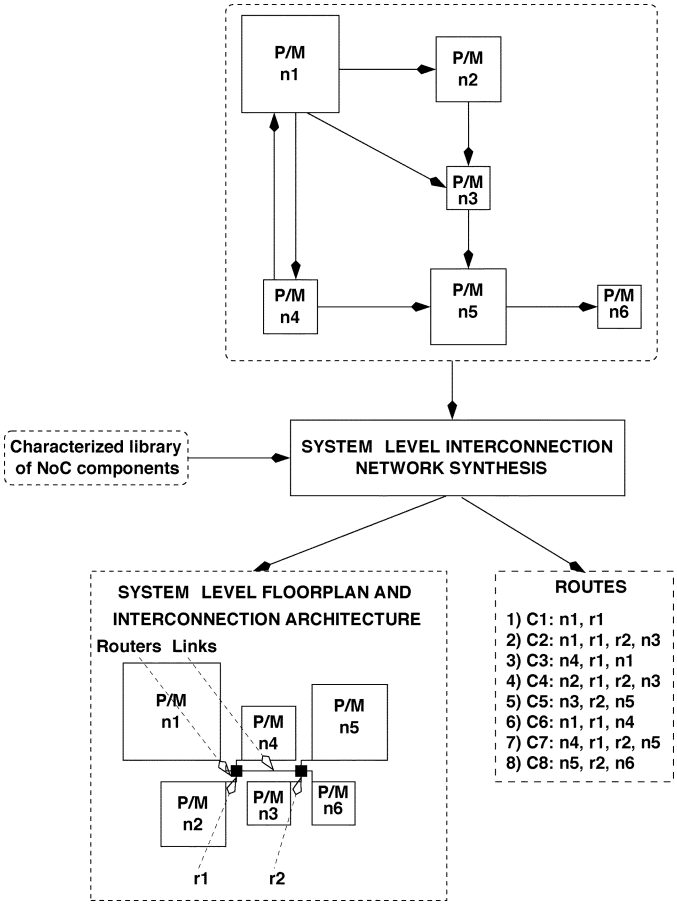

Fig. 1. System-level NoC architecture synthesis.

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 is 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-Mb/s bandwidth requirement. Applications in multimedia and network processing domains demonstrate well-defined periodic communication characteristics and hence, can be easily modelled in the trace graph.[1]

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: 1) the number of router ports; 2) the peak bandwidth supported at input or output ports;[2] and 3) the power consumed to transfer data across the ports. The power consumption in a port is a function of the total traffic flowing through it. Hence, the port is characterized by power consumed per unit bandwidth of traffic. 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 flowing through the link and the length of the link. Therefore, the physical links are characterized by power consumed per unit bandwidth per unit length.

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. Calculation 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 Fig. 1. For example, C2 begins from "n1," passes through "r1" and "r2," and ends at "n3."

In the following sections, we first characterize the router that we utilized for generating the experimental results, and then define the custom NoC synthesis problem.

A. Router Power and Performance Characterization

Banerjee *et al.* [9] presented a cycle accurate power and performance model of a generic NoC router architecture that can be incorporated in mesh topologies. The router architecture had five ports, each of which was composed of an input and output port for bidirectional data transfer. We enhanced the router architecture to support custom topologies and application-specific routing scheme. Custom routing is achieved by including a communication trace identifier (ID) in every packet. The ID uniquely identifies a particular data stream flowing from a source node to the destination. On reception of a packet, the header decoder decides the output port based on the ID.

We characterized the power consumption of the input and output ports of a router in 100 nm technology with the help of a cycle accurate power and performance evaluator[3] [9]. In the experiment, the width of the physical links [consequently, the width of input and output (I/O), first-in, first-out (FIFO), and crossbar] is 32 bits, number of virtual channels is 2, the 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 two-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 in a particular direction is 16 bits/cycle or 0.0625 packets/cycle. The clock period was conservatively assumed to be 3 ns. The simulator was first allowed to stabilize for 3 μs (1000 clock cycles), and the data was collected over the next 7 μs (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.

[1]In the case that the application has a wide variation in the bandwidth requirements, the designer can specify either average or worst case communication traffic.

[2]As the NoC is designed after the computation architecture, the traffic bandwidth produced/consumed by a core is known. Since every core is connected to a router, we assume that the input/output ports can support the required bandwidth of each computation element.

[3]Although the following discussion is based on a router architecture assumed by our paper, the characterization techniques, observations, and conclusions are applicable to other router architectures. Consequently, the proposed techniques are applicable to other router architectures as well.

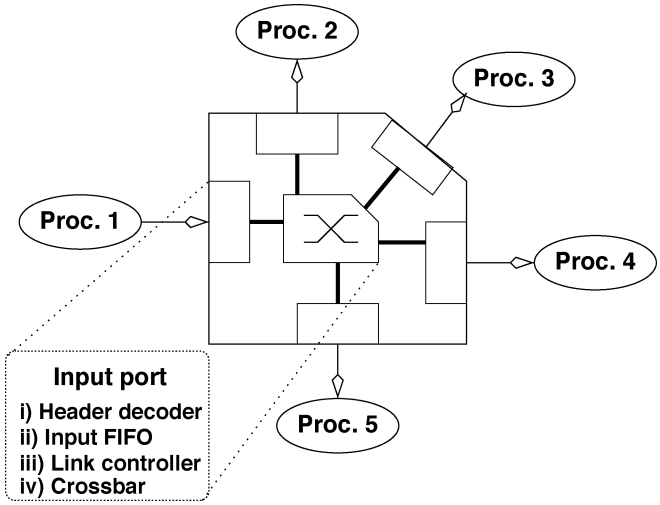

Fig. 2. Input port power consumption setup.

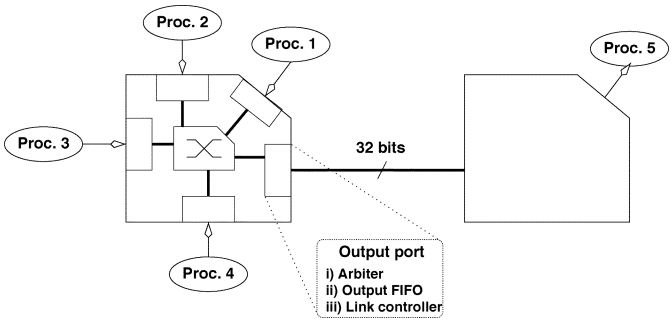

Fig. 3. Output port power consumption setup.

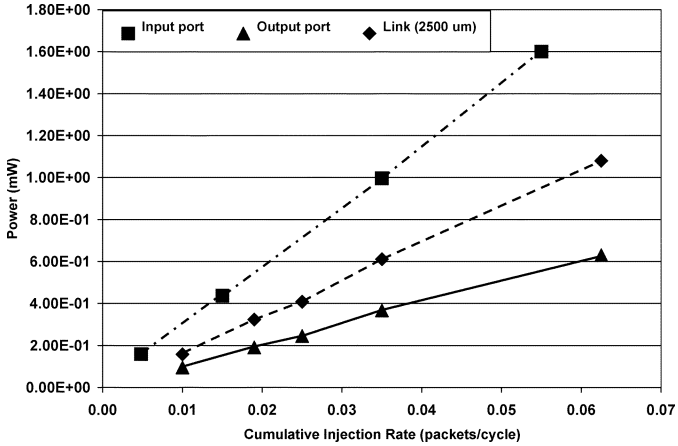

Fig. 4. Power consumption versus injection rate.

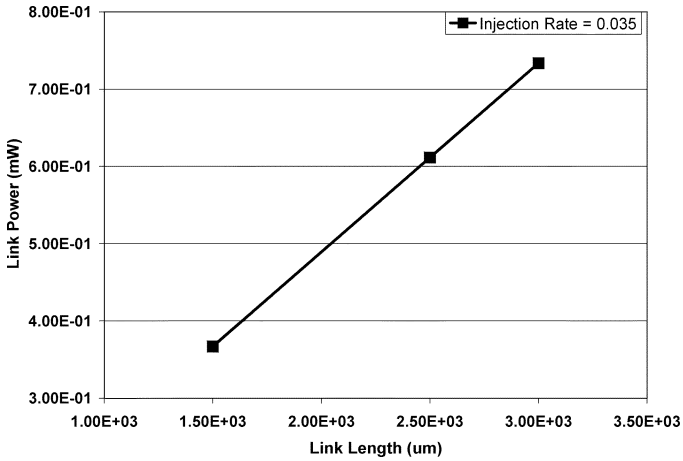

Fig. 5. Link power consumption versus length.

1) Power Consumption in the Input Port: 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 five-port router with five processors attached to each port as shown in Fig. 2. Processor 1 sends packets with random contents to the four other processors with a uniformly random distribution. Fig. 4 plots the power consumption in the input port 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 3 ns, the power per Mb/s of input data passing through the port is given by $P = 328$ nW/Mb/s.

2) Power Consumption in the Output Port: The power consumption at the output port is the summation of the power consumed by the arbiter, output FIFO, and link controller. Again, we considered a five-port router with processors (1–4) attached to four input ports as shown in Fig. 3. All four ports inject packets that traverse through the fifth port to processor 5 attached to the neighboring router. The contents of the packets were randomly generated, and the delay for a particular injection rate was also uniformly distributed within the mean delay interval. Fig. 4 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 packets/cycle passing through the port and 5.6 is the slope of the line. Again, for a clock cycle of 3 ns, the power/Mb/s of output traffic passing through the port is given by $P = 65.5$ nW/Mb/s.

3) Power Consumption in Physical Links: Figs. 4 and 5 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 0.035 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 Fig. 4 can be approximated as $(16.915 \times i)$ mW, where i is the injection rate in packets/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 3 ns, the power consumption of the link per Mb/s/mm can be expressed as $P = 79.6$ nW/Mb/s/mm.

4) Latency: Fig. 6 plots the average latency versus the injection rate for the packets in experiment 2 (Fig. 3). 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

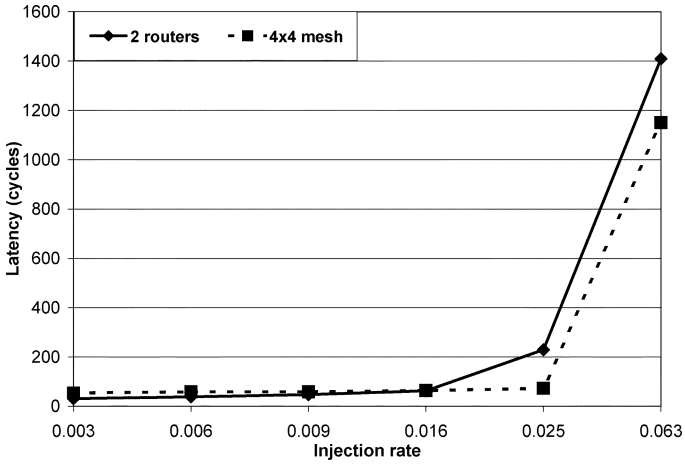

Fig. 6. Latency versus injection rate.

as in Fig. 4. As can be observed from the figure, the average latency remains constant until the output port becomes congested. The injection rate at which the output port becomes congested is given by 0.01563 packets/cycle/port. A similar trend is observed (see Fig. 6) when we consider a 4×4 mesh architecture with 16 processors that are injecting to uniformly distributed random destinations. 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 uncongested 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).

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/cycle, and $\sigma(e_k)$ denotes the latency constraint in hops;
- a router architecture, where η denotes the number of I/O ports of the router,[4] 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 Mb/s 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 Mb/s/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 $\gamma_{\min}$ and $\gamma_{\max}$, 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 $\gamma_{\min}$ and $\gamma_{\max}$;
- the total system-level power consumption for inter-core communication is minimized.

The floorplanning subproblem is a variation of a quadratic assignment problem [10], and is known to be NP hard. The interconnection network generation problem is a variation of the generalized steiner forest problem [11], 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.

We discuss optimal MILP formulations for both problems.[5] 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 designs 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 select routers [14].

This 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, and, finally, Section VI concludes the paper.

II. Previous Work

In recent years, a number of researchers have proposed architectures, performance evaluation techniques, and optimization

[4]Heterogeneous routers with variable number of ports can also be supported. We can initially design the NoC with the router with maximum ports. Once the topology has been generated, routers with unutilized ports can be replaced with lower port routers.

[5]This paper is an integrated and extended version of our two conference papers [12] and [13].

approaches for NoC. The research presented in this paper falls in the category of automated optimization approaches. Vellanki *et al.* [15] presents a detailed discussion about existing NoC architectures and performance evaluation approaches. Existing research in these two categories has predominantly concentrated on regular mesh-based NoC architectures. In this paper, we present automated techniques to generate application specific NoCs that do not conform to a regular topology. In the following paragraph, we discuss existing automated NoC design approaches.

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. Pinto *et al.* [16] 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.* [17] and Ascia *et al.* [18] presented genetic algorithm-based techniques for mapping tasks on mesh-based NoC architectures. Hu *et al.* [19] presented a branch and bound technique to map IPs onto a regular mesh-based NoC architecture. In [20], the authors extended the work presented in [19] to incorporate a deadlock free deterministic routing function. In [21], the authors presented an algorithm that schedules both computation and communication transactions onto mesh-based NoC architectures under real-time constraints. In all these 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 route the communication traces on the topology, such that the performance constraints are satisfied and the communication power consumption of the NoC is minimized.

III. MILP Formulations

In this section, we present our MILP formulations for the NoC design problem. We address the problem by splitting it into two subproblems: 1) system-level floorplanning with an objective of minimizing the power consumption of the NoC subject to the layout constraints and 2) NoC topology and route generation again with an objective of minimizing the power consumption subject to the performance constraints.[6]

A. System-Level Floorplanning

We present a 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

$$\alpha \cdot \left[\sum_{\forall e(u,v) \in E} \mathrm{dist(u, v)} \cdot \Psi_1 \cdot \frac{\omega(\mathrm{e})}{\sigma^2(\mathrm{e})} \right] + \beta \cdot [X_{\max} + Y_{\max}]$$

where $\mathrm{dist(u, v)}$ 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 coordinates 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.[7]

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 coordinate 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 coordinates of the top-right corner of the placed cores, and $\mathcal{DY}_{i,j}$ denote the corresponding difference between the y coordinates. 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 unrestricted variables.

[6]We also address router utilization by: 1) eliminating redundant routers at floorplanning stage and 2) utilizing minimum number of routers to route each trace at the interconnection architecture design stage.

[7]MILP formulations for system level floorplanning have been proposed in the literature [22]–[24]. We include our formulation so that we can discuss extensions for mesh-based topologies. The floorplanning formulation 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. Existing formulations can be combined with the proposed cost function to generate NoC specific layouts.

- 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 coordinates.

2) Objective Function: The objective function for the floorplanning stage is to

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

where

$$\mathcal{P} = \alpha \cdot \left(\sum_{\forall e(v_i,v_j) \in E} \Psi_l \cdot \frac{\omega(e)}{\sigma^2(e)} \cdot (|\mathcal{D}\mathcal{X}_{i,j}| + |\mathcal{D}\mathcal{Y}_{i,j}|) \right)$$
$$\mathcal{A} = \beta \cdot [X_{\max} + Y_{\max}].$$

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

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

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

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

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

$$\mathcal{D}\mathcal{X}_{i,j} + \mathcal{D}\mathcal{X}_{j,i} + \mathcal{D}\mathcal{Y}_{i,j} + \mathcal{D}\mathcal{Y}_{j,i} \geq 1.$$

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

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

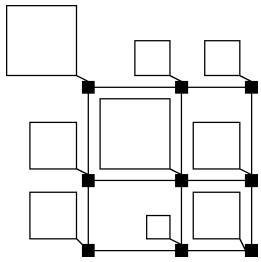

Fig. 7. Example mesh-based floorplan.

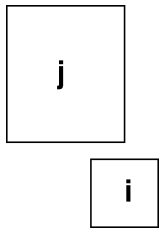

Fig. 8. Illegal layout in mesh-based topology.

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

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

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 Fig. 7. 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 Fig. 8. In Fig. 8, 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 a pair of binary variables $\mathcal{G}\mathcal{X}_{i,j}$ and $\mathcal{G}\mathcal{Y}_{i,j}$ defined as

$$\mathcal{G}\mathcal{X}_{i,j} = \begin{cases} 1, & \text{if } X_{i,\max} > X_{j,\max}; \\ 0, & \text{otherwise.} \end{cases}$$
$$\mathcal{G}\mathcal{Y}_{i,j} = \begin{cases} 1, & \text{if } Y_{i,\max} > Y_{j,\max}; \\ 0, & \text{otherwise.} \end{cases}$$

$\mathcal{G}\mathcal{X}_{i,j}$ and $\mathcal{G}\mathcal{Y}_{i,j}$ 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 $v_i, v_j \in V$:

$$\mathcal{G}\mathcal{X}_{i,j} + \mathcal{X}'_{i,j} \leq 1$$
$$\mathcal{G}\mathcal{Y}_{i,j} + \mathcal{Y}'_{i,j} \leq 1.$$

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

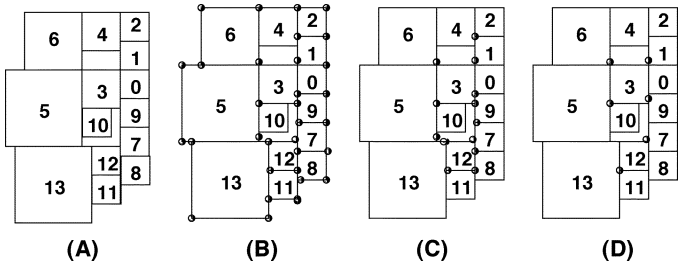

Fig. 9. Example of router allocation for custom topology.

```
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. 10. Algorithm for removing redundant routers.

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 about each other. For example, in Fig. 9(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 coordinates of the respective nodes.

The second step is the placement of the routers. We exploit the fact that the dimensions of routers are much smaller than that of the processing cores [4]. Therefore, once the bounding boxes are generated, we place routers at the nodes of the channel intersection graph [22] 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 Fig. 9(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;
- 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 Fig. 10. First, the algorithm removes the routers along the perimeter of the floorplan. Fig. 9(c) depicts the 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 Fig. 9. 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. Fig. 9(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 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. 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 the minimization 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.

Sections III-B1–III-B3 discuss the variables, objective function, and constraints, respectively.

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 $p_{i,j}$ and $0 \leq j < \eta$ represent the jth 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 nonlinear equation can be easily linearized by the following rule:

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

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 of all the routers. The objective function can be expressed as follows:

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

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/Mb/s 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/Mb/s, $\mathcal{RD}_{k,m}$ denotes the distance between routers k and m, and $\mathcal{ND}_{i,k}$ denotes the distance of node i from router k.

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

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

- *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

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

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, $\forall\{v_i, v_j\} \in E\, \forall v_m \in V, m \neq i, m \neq j, \forall p_{k,l}$

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

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 $\forall(v_i, v_j) \in E$,

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

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 the 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 1 simultaneously. Similarly, $O_{i,j,m,n}$ and $O_{i,j,k,l}$ cannot be 1 simultaneously. Thus, for each pair of routers $\{r_k, r_m\}$, $k \neq m$, $\forall(v_i, v_j) \in E$, $\forall p_{k,l}$, $\forall p_{m,n}$

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

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 router hops that is allowed to route the traffic from a source node to a sink node. 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 sink nodes. In Fig. 11, 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 Fig. 11(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 Fig. 11(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 Fig. 11(c), introduction of a router at a leaf node is equivalent to routing traffic from source

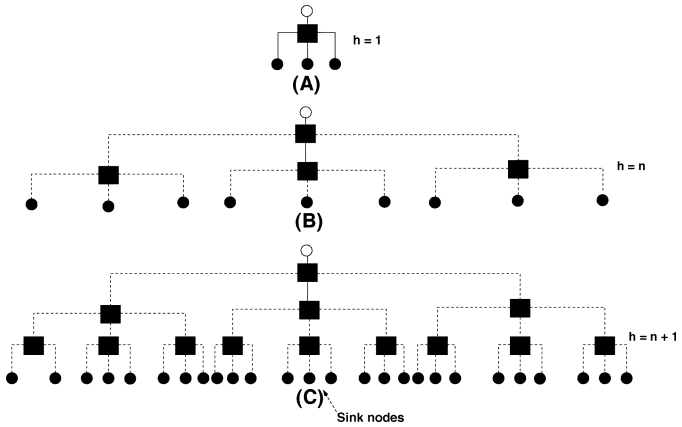

Fig. 11. Adding router to increase number of traffic.

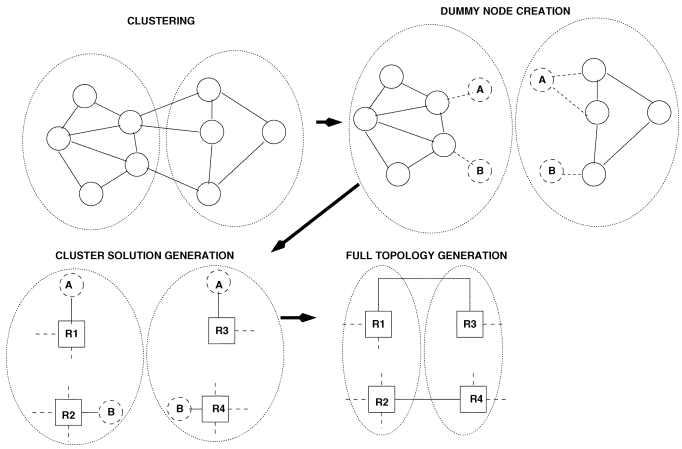

Fig. 12. Clustering based approach.

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

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. We present a clustering-based heuristic technique for reducing the solution times. The heuristic technique is executed after the layout has been generated. The overall approach is shown in Fig. 12.

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 which is specified by the designer. We utilize an algorithm by Johnson *et al.* [25], to form our clusters.[8] For each edge $e \in E$, the clustering algorithm assigns a distance metric to the edge given by $\mathcal{DF}_e = \sigma_e^2/\omega_e$. 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.

[8]http://www.analytictech.com/networks/hiclus.htm

TABLE I
Graph Characteristics

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

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 Fig. 12 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 Fig. 12, 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

The section presents and analyzes the NoC designs synthesized by our techniques for benchmark applications.

A. Experimental Setup

We generated custom NoC architectures for four multimedia benchmarks, namely, mp3 audio encoder, mp3 audio decoder, H.263 video encoder, and H.263 video decoder algorithms. Additionally, 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 Hu *et al.* [19] .

We obtained results for router architectures with five and four 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 328 nW/Mb/s and 65.5 nW/Mb/s, respectively. The link power consumption was estimated to be 79.6 nW/Mb/s/mm. We utilized the Xpress-MP optimizer [27] to solve the MILP problems. The solver was configured with a timeout of 8 h for the floorplanning stage, and a timeout of 12 h for the NoC architecture generation stage. If the solver failed to

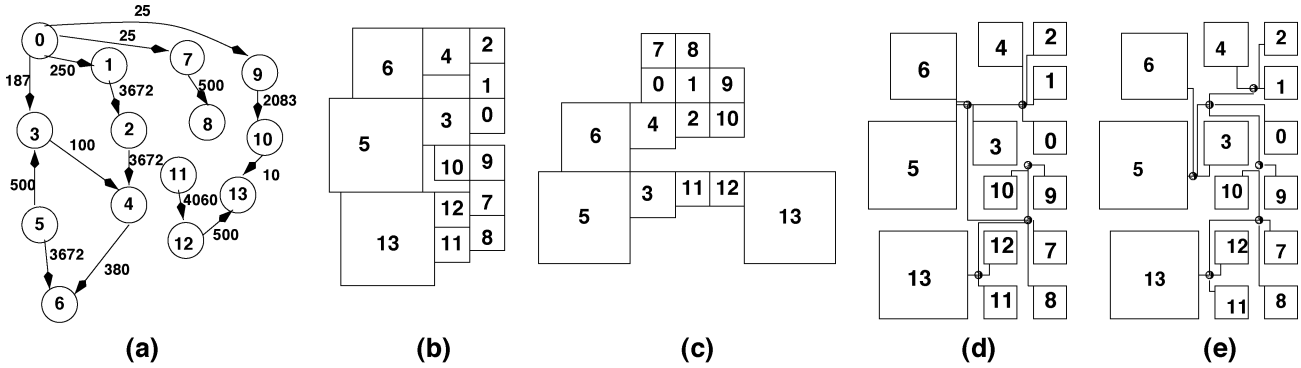

Fig. 13. 263 dec mp3 dec: Communication trace graph, layouts, and NoC designs. (a) Communication trace graph. (b) Custom layout. (c) Mesh layout. (d) NoC with 5-port routers. (e) NoC with 4-port routers.

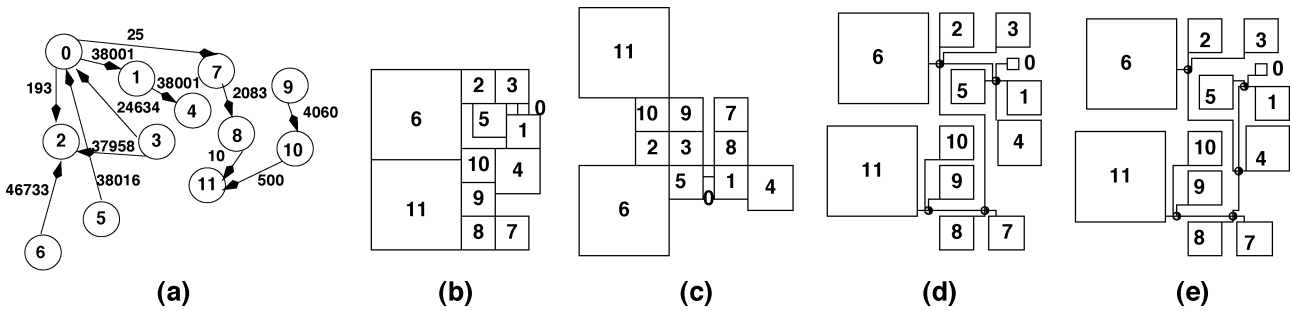

Fig. 14. 263 dec mp3 dec: Communication trace graph, layouts, and NoC designs. (a) Communication trace graph. (b) Custom layout. (c) Mesh layout. (d) NoC with 5-port routers. (e) NoC with 4-port routers.

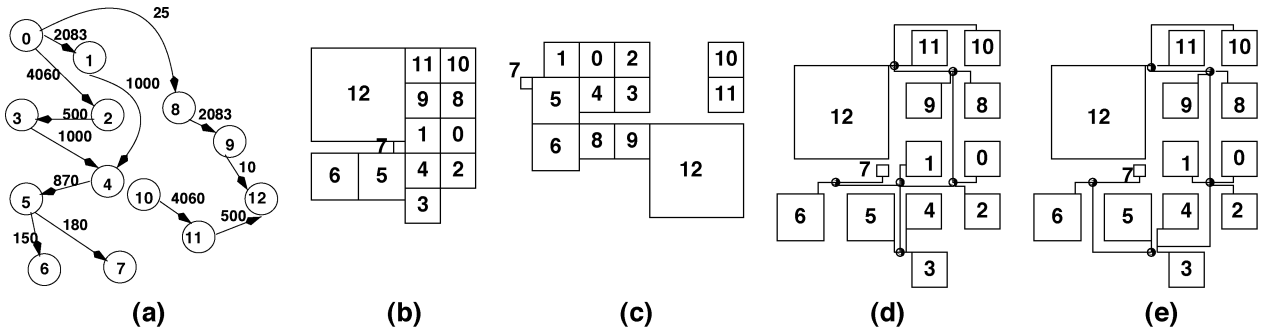

Fig. 15. Mp3 enc mp3 dec: Communication trace graph, layouts, and NoC designs. (a) Communication trace graph. (b) Custom layout. (c) Mesh layout. (d) NoC with 5-port routers. (e) NoC with 4-port routers.

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 nine nodes for the clustering-based heuristic.

All results were obtained on a 950-MHz SPARC processor.

B. Results for Floorplanning Stage

Figs. 13–15 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 kb/s. The node descriptions are depicted in Table II. Figs. 13–15 also present the corresponding floorplans obtained by executing our MILP based floorplanner

TABLE II
NODE DESCRIPTIONS

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 III
RESULTS FOR FLOORPLANNING

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)

on the benchmarks for custom architectures and mesh topologies. The floorplanner places high-communicating nodes close to each other. For example, in the custom layout of Fig. 13, 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. 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 h. The longer running time for mesh may be attributed to the extra constraints required in the formulation to generate mesh topologies.

C. Results for Topology Design and Routing Stage

Table IV compares the results obtained for the MILP- and clustering–based techniques, respectively. In the table, 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 denote theoretical lower bound on the power consumptions of the MILP- and clustering-based formulations, respectively. Columns 10 and 11 give the ratio of the power consumption of the solution of the clustering-based technique to the MILP and clustering lower bounds, respectively. Columns 12 and 13 present the router requirements of MILP and clustering techniques, respectively. Column 14 denotes the ratio of routers required by the clustering solutions over the MILP solutions, and finally columns 15 and 16 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 h.

The MILP-based technique did not generate an optimal result in 12 h for larger sized graphs. However, the clustering-based heuristic is able to generate results in almost all cases (3 exceptions out of 20) in a reasonable amount of time.[9] 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%). The custom topologies of the three benchmarks produced by clustering-based techniques for five-port and four-port routers are shown in Figs. 13–15.

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 that 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 compared the custom NoC designs against solutions with mesh-based NoC topologies. We estimated 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 quality-of-service NoC (QNoC) architecture [28]. 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. Table V shows the results of the comparative study. The number of ports in the routers was five. 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 predesigned physical connections in both the mesh-based topologies force the communication traces to pass through more routers, thus, leading to the increased power consumption and latency. This advantage would diminish as the proportion of power consumption of the physical links to the power consumption of the routers increases. In the comparative study, we focused on 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.

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

[9]The experiments utilized a cluster size of nine 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.

TABLE IV
COMPARISON OF MILP- AND CLUSTERING-BASED TECHNIQUES

No.	No. Ports	Graph	No of Clusters	Power (μW)			Power lower bound (μW)				Routers			Runtime (secs)	
				MILP (M)	Cluster (C)	Ratio C/M	MILP (ml)	Cluster (cl)	Ratio C/ml	Ratio C/cl	MILP (MR)	Cluster (CR)	Ratio CR/MR	MILP	Cluster
1	5	G1	1	2.622	2.622	1	2.622	2.622	1	1	1	1	1	< 1	< 1
2	5	G2	1	108.3	108.3	1	108.3	108.3	1	1	2	2	1	330	330
3	5	G3	1	5.7	5.7	1	5.7	5.7	1	1	2	2	1	1720	1720
4	5	G4	1	5.722	5.722	1	5.722	5.722	1	1	3	3	1	2318	2318
5	5	G5	2	179.5	110.9	0.61	93.26	110.9	1.18	1	5	4	0.8	41200 (t.o)	1103
6	5	G6	2	8.635	8.157	0.94	5.50	8.15	1.48	1	5	5	1	41200 (t.o)	2099
7	5	G7	2	11.91	8.535	0.71	8.10	8.53	1.05	1	5	5	1	41200 (t.o)	35522
8	5	G8	2	170.7	155.2	0.90	116.9	155.2	1.32	1	5	5	1	41200 (t.o)	1559
9	5	G9	2	245.4	115.6	0.47	90.74	115.6	1.27	1	7	5	0.7	41200 (t.o)	35467
10	5	G10	2	15.52	11.54	0.74	10.97	11.54	1.05	1	7	6	0.8	41200 (t.o)	1540
11	4	G1	1	2.631	2.631	1	2.631	2.631	1	1	2	2	1	< 1	< 1
12	4	G2	1	138.0	138.0	1	138.0	138.0	1	1	4	4	1	347	347
13	4	G3	1	5.944	5.944	1	5.944	5.944	1	1	3	3	1	1206	1206
14	4	G4	1	5.722	5.722	1	5.722	5.722	1	1	4	4	1	22222	22222
15	4	G5	2	194.6	140.7	0.72	121.4	140.7	1.15	1	5	5	1	41200(t.o)	2502
16	4	G6	2	10.94	12.47	1.12	8.20	10.4	1.52	1.19	6	6	1	41200(t.o)	41200(t.o)
17	4	G7	2	13.90	8.664	0.62	6.88	8.66	1.25	1	6	6	1	41200(t.o)	36733
18	4	G8	2	158.6	209.4	1.31	102.1	134.3	2.05	1.55	7	7	1	41200(t.o)	41200(t.o)
19	4	G9	2	241.3	147.2	0.61	80.33	113.7	1.83	1.29	7	7	1	41200(t.o)	41200(t.o)
20	4	G10	2	17.22	11.97	0.69	9.01	11.97	1.32	1	8	8	1	41200(t.o)	36544

TABLE V
COMPARISON OF CUSTOM (C), MESH (M), AND QNOC (Q) DESIGNS

Graph	Power (μW)					Routers				
	C	M	Ratio M/C	Q	Ratio Q/C	C	M	Ratio M/C	Q	Ratio Q/C
G1	2.622	7.363	2.80	2.627	1.00	1	5	5	2	2
G2	108.3	291.4	2.69	216.1	1.99	2	7	3.5	4	2
G3	5.7	10.51	1.84	6.368	1.11	2	8	4	4	2
G4	5.722	12.51	2.18	6.453	1.12	3	9	3	3	1
G5	110.4	273.7	2.47	244.9	2.21	4	12	3	6	1.5
G6	8.157	18.02	2.21	12.35	1.51	5	13	2.6	5	1
G7	8.535	22.27	2.60	22.37	2.62	5	14	2.8	6	1.2
G8	155.2	277.0	1.78	155.7	1.00	5	15	3	5	1
G9	115.6	296.7	2.56	352.4	3.04	5	16	3.2	7	1.4
G10	11.54	28.63	2.15	25.86	1.95	6	17	2.8	7	1.1

total of 20) of the cases, we had some ports that supported 3 traffic traces. Warnakulasuriya *et al.* [29] 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.[10]

VI. CONCLUSION

We defined the application-specific NoC synthesis problem and proposed linear programming based solutions. We addressed the complexity of the NoC synthesis problem by dividing it into two stages: floorplanning and interconnection network generation. We presented optimal MILP formulations for the two stages and presented a low run time clustering-based heuristic for the second stage. 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 designs generated by the clustering technique consumed only 85% of the power and 96% of the router resources, respectively, compared to the MILP formulation. In comparison to the NoC designs synthesized by our technique, 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, respectively.

Technology scaling will result in an increase in the static power consumption of routers and dynamic power consumption of the physical links. Our techniques account for the dynamic power consumption due to the physical links. Static power consumption can be minimized by removing router ports that do not support any traffic. Virtual channels are chief contributors of leakage power consumption [30]. Their contribution can be reduced by over 80% by deploying run time power management schemes [30].

Our techniques do not currently address fault-tolerant topologies. Automated design techniques for fault-tolerant topologies would be the 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. Our paper addresses the NoC design in isolation. Integration of computation architecture design with NoC synthesis has the potential for larger power savings.

[10]In general, the final number of virtual channels may vary. Increasing the number of virtual channels leads to diminishing performance improvements. As the power consumption is directly influenced by the performance, the characterization figures for a router with a fixed number of virtual channels can be utilized for NoC design.

REFERENCES

[1] International Technical Roadmap for Semiconductors [Online]. Available: http://public.itrs.net/ 2004

[2] D. Sylvester and K. Keutzer, "A global wiring paradigm for deep submicron design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 2, pp. 242–252, Feb. 2000.

[3] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.

[4] W. J. Dally and B. Towles, "Route packet, not wires: On-chip interconnection networks," in *Proc. Des. Automat. Conf.*, 2002, pp. 684–689.

[5] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *IEEE Comput.*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[6] M. B. Taylor *et al.*, "The RAW microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE Micro*, vol. 22, no. 6, pp. 25–35, Mar./Apr. 2002.

[7] A. Jalabert, S. Murali, L. Benini, and G. De Micheli, "XpipesCompiler: A tool for instantiating application specific networks on chip," in *Proc. Des. Automat. Test Eur.*, 2004, pp. 884–889.

[8] G. D. Micheli, R. Ernst, and W. Wolf, *Readings in Hardware/Software Co-Design*. San Mateo, CA: Morgan Kaufmann, 2001.
[9] N. Banerjee, P. Vellanki, and K. S. Chatha, "A power and performance model for network-on-chip architectures," in *Proc. Des. Automat. Test Eur.*, 2004, pp. 1250–1255.
[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
[11] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III, "Approximation algorithms for degree-constrained minimum-cost network-design problems," *Algorithmica*, vol. 31, no. 1, pp. 58–78, 2001.
[12] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear programming based techniques for synthesis of network-on-chip architectures," in *Proc. Int. Conf. Comput. Des.*, 2004, pp. 422–429.
[13] K. Srinivasan and K. S. Chatha, "A methodology of layout aware design and optimization of custom network-on-chip architectures," presented at the Proc. Int. Symp. Quality Electron. Des., San Jose, CA, 2006.
[14] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C-36, no. 5, pp. 547–553, 1987.
[15] P. Vellanki, N. Banerjee, and K. S. Chatha, "Quality-of-service and error control techniques for mesh based network-on-chip architectures," *Integr., VLSI J.*, vol. 38, no. 3, pp. 353–382, Jan. 2005.
[16] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli, "Efficient synthesis of networks on chip," in *Proc. Int. Conf. Comput. Des.*, 2003, pp. 146–150.
[17] T. Lei and S. Kumar, "A two-step genetic algorithm for mapping task graphs to a network on chip architecture," in *Proc. Euromicro Symp. Dig. Syst. Des.*, 2003, pp. 180–187.
[18] G. Ascia, V. Catania, and M. Palesi, "Multi-objective mapping for mesh-based NoC architectures," in *Proc. ISSS-CODES*, 2004, pp. 182–187.
[19] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NOC architectures under performance constraints," in *Proc. ASP-Des. Automat. Conf.*, 2003, pp. 233–239.
[20] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proc. Des. Automat. Test Eur.*, 2003, pp. 688–693.
[21] ——, "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints," in *Proc. Des. Automat. Test Eur.*, 2004, pp. 234–239.
[22] S. M. Sait and H. Youssef, *VLSI Physical Design Automation: Theory and Practice*. New York: McGraw-Hill, 1994.
[23] P. Chen and E. S. Kuh, "Floorplan sizing by linear programming approximation," in *Proc. Des. Automat. Conf.*, 2000, pp. 468–471.
[24] J. G. Kim and Y. D. Kim, "A linear programming based algorithm for floorplanning in VLSI design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 5, pp. 584–592, May 2003.
[25] R. D'andrade, "U-statistic hierarchical clustering," *Psychometrica*, vol. 4, no. 1, pp. 58–67, 1978.
[26] Analytic Technologies. [Online]. Available: http://www.analytictech.com/networks/hiclus.htm, 2004
[27] Dash Optimization [Online]. Available: http://www.dashoptimization.com, 2004
[28] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost considerations in network on chip," *Integr. VLSI J.*, vol. 38, no. 1, pp. 19–42, Nov. 2004.
[29] S. Warnakulasuriya and T. M. Pinkston, "Characterization of deadlocks in irregular networks," *J. Parallel Distrib. Syst.*, vol. 62, no. 1, pp. 61–84, 2002.
[30] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnection networks," in *Proc. Int. Symp. Low Power Electron. Des.*, 2003, pp. 90–95.

Krishnan Srinivasan (S'01) received the B.S. degree in electrical engineering from the Regional Institute of Technology, India, in 1999, and the M.S. degree in electrical engineering from Arizona State University, Tempe, in 2002, where he is currently pursuing the Ph.D. degree in computer science and engineering.

His current research focuses on mathematical models and approximation algorithms for low-power system-on-chip (SoC) and network-on-chip (NoC) design. His research interests are in the field of power and performance optimization of SoC and NoC architectures.

Mr. Srinivasan is a student member of the Association for Computing Machinery (ACM).

Karam S. Chatha (M'01) received the B.E. degree (with honors) in computer technology from Bombay University, Mumbai, India, in 1993 and the M.S and Ph.D. degrees in computer science and engineering from the University of Cincinnati, Cincinnati, OH, in 1997 and 2001, respectively.

He is currently an Assistant Professor in the Department of Computer Science and Engineering at Arizona State University, Tempe. 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.

Dr. Chatha is a recipient of the National Science Foundation's CAREER Award in 2006. He received the "Best Paper Award" at the International Workshop on Field Programmable Logic (FPL) in 1999. He is a member of the Association of Computing Machinery (ACM).

Goran Konjevod received the B.S. degree in mathematics from the University of Zagreb, Croatia, in 1995, and the M.S. degree in algorithms, combinatorics, and optimization, and the Ph.D. degree in applied mathematics from Carnegie Mellon University, Pittsburgh, PA, in 1998 and 2000, respectively.

He is currently an Assistant Professor in the Department of Computer Science and Engineering at Arizona State University, Tempe. 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. He presented (jointly with R. D. Carr of Sandia National Laboratories) a tutorial on Polyhedral Combinatorics at the 2004 INFORMS meeting in Denver.

Dr. Konjevod is a member of the Society for Idustrial and Applied Mathematics (SIAM) and the Mathematical Association of America (MAA).