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

Krishnan Srinivasan, Karam S. Chatha and Goran Konjevod
Department of Computer Science and Engineering, Arizona State University, Tempe, AZ, 85287-5406.

Abstract

Network-on-chip (NoC) has been proposed as a solution for the communication challenges of System-on-chip (SoC) design in the nanoscale regime. 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. In the nanoscale regime, low power consumption would continue to be an important design goal. We first discuss an optimal mixed integer linear programming (MILP) formulation that synthesizes a low power NoC architecture subject to the performance constraints. The MILP formulation is limited by large run times. We next present heuristic techniques that exploit clustering, and 0-1 constraint relaxation to reduce the run times of the formulation. The techniques minimize power as the primary goal, and minimize the number of routers (area) as a secondary goal. We present an analysis of the quality of the results and the solution times of the proposed techniques by extensive experimentation with the realistic benchmarks. The clustering based heuristic generates results whose power consumption is within 11 % of the MILP solutions and its average run time is 171.1 seconds. The average run time of the relaxation and rounding based techniques is less than 2 seconds, and the power consumption of their solutions is within 58 % of the MILP result.

1 Introduction

Network-on-Chip (NoC) has been proposed as a solution for the communication challenges in the nanoscale regime [1] [2]. Packet switching supports asynchronous transfer of information. It provides extremely high bandwidth by distributing the propagation delay across multiple switches, and thus pipelining the signal transmission. In the right half of Figure 1, an SoC architecture with an NoC is depicted. In the figure, the various "P/M" blocks denote processing (DSP, ASIC, FPGA) cores or storage elements (Cache, SRAM, CAM), and "R" denotes the router nodes. The lines between various blocks represent the network links. The "R" blocks along with the physical links form the NoC.

The computer-aided design of an NoC based application specific SoC architecture is an open problem. Automated design needs to solve two key problems: (i) computation architecture synthesis and mapping (that has been addressed widely in the system-level synthesis community), and (ii) communication architecture synthesis and mapping. 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. *Application specific communication architecture synthesis and mapping is the focus of this paper.*

Communication architecture synthesis is shown in Figure 1. The input to the communication architecture synthesis problem is the computation architecture specification, characterized interconnection network elements, and performance constraints. The computation architecture specification consists of processing and memory elements shown by rectangular blocks labeled "P/M" in the top left corner of the figure. The directed edges between two blocks represent the communication traces. The communication traces are annotated as "C*m*(B,L)" where '*m*" represents the trace number, "B" represents the bandwidth requirement, and "L" is the latency constraint. The interconnection router architecture specification is shown on the top right hand side corner of the figure. The router specification consists of i) number of ports, ii) maximum bandwidth that can be supported at any one port, iii) latency, and iv) power consumption for data forwarding. The output of the communication architecture synthesis problem is a topology of the network, and mapping or static routing of the communication traces on the network such that the performance constraints are satisfied, and the power consumption is minimized. The topology of the network specifies the number of routers, and their interconnections. The static routing of a communication trace is shown by the annotation of physical links in the bottom half of Figure 1. For example , C2 begins from "P/M1", passes through "R1" and "R2", and ends at "P/M3". After the NoC architecture has been synthesized, elimination of possible deadlocks between the communication traces can be achieved by introduction of additional virtual channels in the routers as a post-processing step [3]. The system-level design along with the NoC architecture is then input to a floorplanning tool that generates the final layout.

Figure 2 depicts the plots of the average network latency (on left hand side y-axis) and power consumption (on right hand side y-axis) versus the bandwidth (or injection rate) of a 4x4 mesh based NoC architecture in 180 nm technology.

Proceedings of the IEEE International Conference on Computer Design (ICCD'04)
1063-6404/04 $ 20.00 IEEE

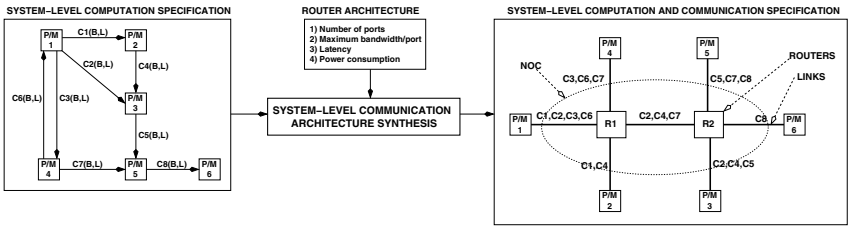

Figure 1. NoC Synthesis

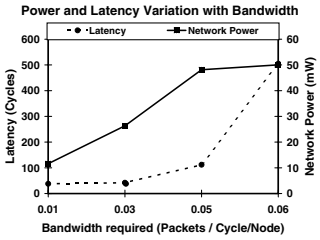

Figure 2. Power and latency variation

The plots were obtained by utilizing a performance evaluator [4]. As can be seen from the plots, increase in bandwidth of the network results in a proportional increase in the power consumption. The power consumption continues to increase till the network gets saturated or congested. Network congestion is marked by a sharp increase in the latency. *Network congestion can be avoided by mapping communication traces such that peak bandwidth is not violated on any router port in the network.* Our synthesis technique prevents network congestion by static routing of the communication traces subject to the peak bandwidth constraint on the router ports. The power consumption in the un-congested region is given by $\mathcal{P} = \mathcal{P}_0 + m * \omega$, where $\mathcal{P}$ is the total power, $\mathcal{P}_0$ is the static (or leakage) power consumption of the router in the absence of traffic, m is the slope, and ω is the bandwidth. $\mathcal{P}_0$ is a constant for a given router architecture. The network latency also remains (more or less) constant until the network is congested. Since, the network is always operated in the un-congested mode, we can represent the network latency in terms of router hops (such as 1 or 2) instead of an absolute number (such as 20 cycles). In the following section we define the performance constrained NoC synthesis problem.

1.1 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 direct edge $e_k = \{v_i, v_j\} \in E$ denotes a communication trace from v_i to v_j.
- For every $e_k = \{v_i, v_j\} \in E$, $\omega(e_k)$ denotes the bandwidth requirement in bits per second, 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 bandwidth (in the un-congested mode) that the router can support on any one port. Since a node $v \in V$ is attached to a port of a router, the cumulative bandwidth to and from any node is less than Ω.
- A constant $\mathcal{R}_{max}$ that refers to the maximum number of routers that can be utilized under the area constraints of the NoC floorplan.

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 obtain a network topology $T(\mathcal{R}, V, E_r, E_v)$ such that:

- $|\mathcal{R}| \leq \mathcal{R}_{max}$,
- 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 that satisfies $\omega(e_k)$, and $\sigma(e_k)$, and
- The total power consumption is minimized.

The NoC synthesis problem is a variation of the generalized steiner forest problem that is known to be NP hard [5]. *As mentioned earlier, the power consumption of the NoC in the un-congested mode varies linearly with the traffic flowing through the network.* Therefore, the power consumption of the NoC can be minimized by minimizing the cumulative traffic flowing through the ports of all routers. *In this paper, we present linear programming based techniques for solving the above problem*. We discuss an optimal mixed integer linear programming (MILP) formulation for NoC synthesis. The MILP formulation is constrained by large solution times. We also present heuristic techniques for alleviating this limitation. The first heuristic technique exploits clustering to reduce the individual MILP problem size. The second heuristic relaxes the 0-1 constraints in the MILP problem to generate a linear programming (LP) relaxation. The fractional optimal solution of the LP relaxation is utilized to generate valid NoC architectures by deterministic and randomized rounding based techniques.

The paper is organized as follows: Section 2 discusses the previous work, Section 3 presents the MILP formulation, Section 4 presents the clustering based approach, Section 5 presents the relaxation and rounding based heuristics, Section 6 presents the experimental results, and finally Section 7 concludes the paper.

2 Previous work

The existing work on NoC design has chiefly concentrated on performance evaluation [4] [6] [7], router architecture design [8] [9], designer specified network instantiation [10], and error control [11] [12]. In contrast, our work addresses the automated synthesis of custom NoC topologies and mapping of the communication traces on the architecture. Pinto

et. al [13] 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. Hu et. al. [14] presented a heuristic technique for computation and communication mapping in a regular tile based NoC architecture. They assume that the NoC architecture already exists, and it has a regular mesh topology. We differ from the above mentioned work in a fundamental aspect. Our work addresses design of an application specific NoC, and does not assume an existing interconnection network architecture. We synthesize a custom NoC architecture, and map (route) the communication traces on the topology such that the performance constraints are satisfied and the communication power consumption is minimized. *To the best of our knowledge, automated low power synthesis of application NoC architectures has not been addressed before.*

3 MILP formulation

In this section, we present our MILP formulation for the NoC synthesis problem. As discussed in Section 1.1, the NoC power consumption can be minimized by minimizing the total traffic flowing through the routers. Therefore, we formulate the MILP problem with an objective function that minimizes the cumulative traffic flow through the ports of the routers, subject to performance constraints. In Section 3.1 we define the variables of the formulation, in Section 3.2 we state the objective function, and finally in Section 3.3, we present the constraints.

3.1 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 router a $r_i \in \mathcal{R}$.
- *Node-to-port mapping variables:* For each node $v_k \in G$, 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 $r_i \in \mathcal{R}$, otherwise 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 denoting 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 denoting 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 denoting 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.

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

- *Variable denoting 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.

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

- *Total bandwidth usage of a port:* Let $\mathcal{B}_{i,j}$ represent the total bandwidth usage of port $p_{i,j}$. The total bandwidth usage of a port is the sum of the traffic entering the port and traffic leaving that port. Therefore, $\mathcal{B}_{i,j}$ can be represented as

$$\mathcal{B}_{i,j} = \mathcal{BI}_{i,j} + \mathcal{BO}_{i,j}$$

3.2 Objective Function

The objective is to minimize the power consumption of the NoC by minimizing the cumulative traffic flowing through ports of all the routers. The objective function can be expressed mathematically as follows:

$$Minimize\ \Sigma_{\forall r_i \in \mathcal{R}} \Sigma_{\forall p_{i,j}} \mathcal{B}_{i,j}$$

3.3 Constraints

The following constraints are formulated.

- *Port capacity constraint:* The bandwidth usage of a port should not exceed its capacity. Therefore,

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

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

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

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,

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

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

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

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

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

  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 $\{r_k, r_m\}$, $k \neq m$, $\forall (v_i, v_j) \in E, \forall p_{k,l}, \forall p_{m,n}$

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

  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 $\forall \{v_j, v_i\} \in E$

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

- *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 modeled as follows:

$$\forall e_k = \{v_i, v_j\} \in E, \Sigma_{\forall r_k \mathcal{R}} \Sigma_{\forall p_{k,l}} \mathcal{O}_{i,j,k,l} \leq \sigma_{e_k}$$

  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 an NoC architectures. 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 $\forall e_k \in E, \sigma(e_k) \leq \sigma$), an NoC topology is not possible if for any node, the total number of edges entering and leaving the node is more than $(\eta - 1)^\sigma$.

4 Clustering based heuristic technique

The MILP formulation is constrained by exponentially increasing solution times for communication trace graphs with

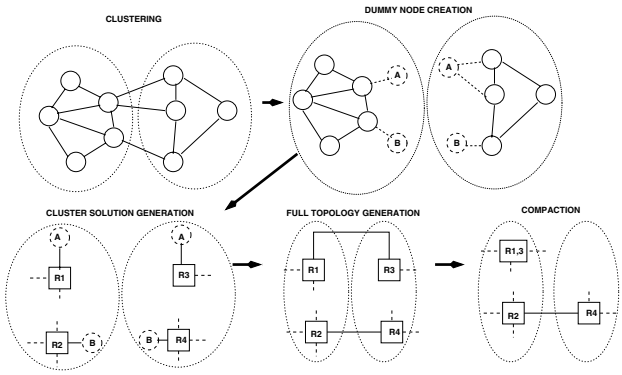

Figure 3. Clustering based approach

large number of edges. This section presents a clustering based heuristic technique for reducing the solution times. The overall approach is shown in Figure 3.

The first stage is to form clusters of nodes. The sizes of the clusters are constrained by the maximum number of communication traces or edges in the clusters. This information is specified by the designer. We utilize an algorithm by Johnson et al. [15] to form our clusters. The clustering algorithm treats the latency constraint as a distance metric. Two communicating nodes that have low latency and therefore are close to each other (distance metric) 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 3 the top two edges that are cut share a common source in the left hand side cluster. Hence, only two dummy nodes (that are labeled "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 is then utilized to generate the partial solution for each cluster. In the figure, we assume that the routers have four ports that 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 3, 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.

Finally, the topology is compacted by comparing neighboring routers that are across clusters. The traffic routed through two neighboring routers that are across clusters, can be compacted or collapsed on a single router if:

- the total number of utilized ports (U) on one router is less than or equal to the number of free ports on the neighboring router (F) plus two ($U \leq F + 2$), and
- the bandwidth constraints on all the ports of the compacted router are satisfied.

For example, in the bottom right hand side of Figure 3 router R1 has 3 ports that are utilized, and router R3 has 2 free ports, and they are compacted into one. In the figure, we have assumed that the bandwidth constraints on the ports are satisfied.

5 Relaxation and rounding based heuristic techniques

The linear programming (LP) formulation is obtained by relaxing all the integer constraints in the MILP formulation. (We also remove some of the traffic routing constraints because they are redundant in the linear relaxation.) An optimal solution to any linear program can be generated in time polynomial in the number of variables and constraints [16]. We obtain valid solutions by the application of rounding based heuristic techniques that replace the fractional values assigned to various variables by integer values. In the following section we first discuss the LP relaxation, and then present deterministic and randomization based rounding heuristics.

5.1 LP formulation

The solution to the LP formulation assigns real values to all the variables. In this situation it is not possible to prevent the splitting of communication traces across multiple ports. Due to this, several of the traffic routing constraints become redundant in the linear relaxation and we remove them. For example, the LP solution may have a communication trace exiting a port of a router, and "0.5" of the trace entering port $p_{1,1}$ of router r_1, "0.25" of the trace entering port $p_{1,2}$ of router r_2, and "0.25" of the trace entering port $p_{1,3}$ of router r_3. Thus, the LP solution permits the splitting and joining of the traffic between ports. The LP constraints are summarized below. In the interest of space, we do not discuss the constraints that were presented earlier in the MILP formulation.

- *Port capacity constraint:* Same as MILP.
- *Node-to-port mapping constraint:* Same as MILP.
- *Traffic routing constraints:*
  - The total traffic entering a router should be equal to that leaving that router. Therefore,

$$\forall r_i \in \mathcal{R}, \Sigma_{\forall j \leq \eta} \mathcal{BI}_{i,j} = \Sigma_{\forall j \leq \eta} \mathcal{BO}_{i,j}.$$

  - In order to ensure the flow of traffic from source to sink, a traffic leaving a port of a router should enter some port of a different router, or enter a sink node. Similarly, a traffic entering a port of a router should have left from some port of a different router or it should have left from a source node. Thus, $\forall (v_i, v_j) \in E, \forall r_k \in \mathcal{R}, \forall p_{k,l}$,

$$\mathcal{O}_{i,j,k,l} \leq \mathcal{NR}_{j,k,l} + \Sigma_{\forall p_{m,n}, m \neq k} \mathcal{I}_{i,j,m,n}$$
$$\mathcal{I}_{i,j,k,l} \leq \mathcal{NR}_{i,k,l} + \Sigma_{\forall p_{m,n}, m \neq k} \mathcal{O}_{i,j,m,n}$$

– Additionally, traffic routing constraints 1 and 4 of the MILP formulation are also included in the LP formulation.

- *Latency constraint:* Same as MILP.

5.2 Rounding based heuristics

In this section, we shall present two rounding based heuristics that generate valid solutions to the NoC synthesis problem. The heuristic techniques incrementally build the NoC topology from the LP solution and map the communication traces on the partial architecture. The techniques take one communication trace at a time and build a path with physical links and routers for mapping the selected traffic. Thus, the topology generation and mapping of communication trace occur in an integrated manner. In the following sections, we first discuss the priority functions, and then present the two rounding based heuristic techniques.

5.2.1 Heuristic priority function

Since the node-to-port mapping variable is assigned a fractional value in the LP solution, the first step is to select a node (source or sink of a communication trace) and assign it to one port of a router. Further, the construction of the topology also involves selection of a communication trace that will be mapped. These selection decisions are both influenced by the bandwidth and latency constraints. A communication trace with a low value for latency, and a high bandwidth requirement allows only restricted design space exploration. On the other hand, an edge with a high value for latency and low bandwidth requirement permits liberal design space exploration. The bandwidth constraint can be satisfied by the addition of extra routers. The latency constraint on the other hand, cannot be adhered to by the addition of routers. Therefore, latency is a more constraining parameter compared to bandwidth. We propose a heuristic that combines both latency and bandwidth to select nodes and communication traces during the execution of the technique. The heuristic for selecting a node u is defined as $\mathcal{H}(u) = \Sigma_{\forall(u,v)\in E}\frac{\omega(u,v)}{\sigma(u,v)^2} + \Sigma_{\forall(v,u)\in E}\frac{\omega(v,u)}{\sigma(v,u)^2}$. The heuristic for selecting an edge $e \in E$ is defined as $\mathcal{H}(e) = \frac{\omega(e)}{\sigma(e)^2}$.

5.2.2 Path based deterministic heuristic (PDH)

The pseudo code for the path based deterministic heuristic is shown in the Figure 4. The inputs to the procedure are: i) solution of LP formulation (*lp_soln*), ii) communication trace graph ($G(V, E)$), iii) router architecture constraints (η, Ω), iv) maximum number of routers ($\mathcal{R}_{max}$), and topology and mapping information (T). Initially, T only includes the routers. T is updated as the links are added and communication traces are mapped.

The first step in the algorithm is to map the nodes (sources or sinks) to ports of the routers present in the partial solution.

```
PDH (lp_soln, G(V,E), η, Ω, R_max, T)
  assign_nodes()
  for (i = 0; i < |E|; i++) /* Build topology and map traces */
    e(u,v) = select(E), r_u = get_router(u), r_v = get_router(v)
    if (r_u = r_v) continue
    p_u = get_free_port(r_u)
    if (p_u = null) add_relay(u, r_u, p_u)
    p_v = get_free_port(r_v)
    if (p_v = null) add_relay(v, r_v, p_v)
    T = connect_map(p_u, p_v, e)
  endfor
  L = sort_descending(T) /* Compaction */
  while (L ≠ null)
    R = get_head(L), R_n = max_neigh(R)
    while (U(R) ≤ F(R_n) + 2 AND bandwidth_satisfied())
      R = compact(R, R_n), L = L − R_n, R_n = max_neigh(R)
    endwhile
    L = L - R
  endwhile
  if (constraint_satisfied(T)) return(T)
  return(FAILURE)
end
```

Figure 4. Path based deterministic heuristic

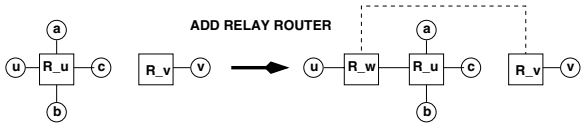

Figure 5. Relay router addition

In the pseudo code the function *assign_nodes()* performs this task. A node is selected based on the heuristic function $\mathcal{H}$ defined above. A node with a larger $\mathcal{H}$ value is given priority. The selected node is mapped to a port $p_{i,j}$ such that i) $p_{i,j}$ is free, and ii)$\mathcal{NR}_{u,i,j}$ has a maximum value among all free ports. In case no free ports are found, the procedure adds a router to the topology (T), and assigns a port of that router to the node. If the addition of a router violates the $\mathcal{R}_{max}$ constraint, the procedure declares failure. The function *assign_nodes()* terminates when all nodes have been assigned a unique port.

In the next phase of the technique, the topology of the NoC is defined, and the communication traces are mapped. In this phase, the least constraining communication trace $e(u, v)$ is selected by invoking *select(E)* that returns an unmapped trace with the smallest $\mathcal{H}$ value. Hence, the less constrained communication traces are mapped first, and the most constrained traces are mapped last. Next the routers of the source (r_u) and sink nodes (r_v) are obtained by calling *get_router()*. If the source (u) and sink (v) node are mapped to the same router, then the technique skips the remaining portion of the current iteration. Alternatively, the technique obtains a free port on the source (p_u) and sink (p_v) routers, respectively. If a free port does not exist on either the source or sink router, a relay router is added by invoking the function

add_relay(). The relay router addition operation is shown in Figure 5 for two 4-port routers. In the figure, the source router R_u does not have any free ports. Hence, a relay router, R_w, is added to the source side that generates free ports. The technique then invokes the *connect_map()* function to attach the free ports by addition of a physical link, map the trace on the path, and update the topology T. Since, the technique builds a solution by addition of relay routers it can lead to increase in latencies of already mapped traces. Hence, the least constraining edges with loose latency constraint (lower $\mathcal{H}$ value) are mapped first, and the most constraining edges with tight latency constraints are mapped towards the end.

After the completion of the topology generation and mapping phase the technique enters the compaction phase. The compaction phase is identical to the one applied in the clustering based heuristic, and is depicted as the last stage in Figure 3. In contrast to the compaction in the clustering based heuristic, the path based heuristic considers any two neighboring routers. The compaction heuristic invokes $sort\_descending()$ to sort the routers in the descending order of their free ports and generates a list L. The router R at the head of the list is selected first, and its neighboring router R_n with maximum number of free ports is obtained by calling $max\_neigh()$. If the compaction conditions are satisfied the two routers are collapsed into R, and the neighbor with maximum number of free ports is selected again. The inner loop continues till the technique can find neighboring routers that can be compacted. The outer loop ends once L is empty. The compaction phase reduces the area and power consumption overhead. Further, it can also lead to possible reduction in the latency of some communication traces. At the end of the compaction phase the technique checks for performance constraint satisfaction, and returns the topology if they are satisfied.

Time complexity: The maximum number of routers that can be generated for the LP solution is R_{max}. The total number of iterations that are performed by the first "for" loop are E, and each iteration takes a constant time. The sort function is of complexity $O(R_{max} log R_{max})$. The total number of compactions that can be performed is $O(R_{max} - 1)$. Hence, the complexity of the PDH heuristic is $O(E + R_{max} log R_{max})$.

5.2.3 Path based randomization heuristic (PRH)

The path based randomization heuristic (PRH) differs from PDH in the node to port assignment (*assign_nodes()*). The PDH heuristic assigns a node u to port $p_{i,j}$ if $\mathcal{NR}_{u,i,j}$ is maximum. $\mathcal{NR}_{u,i,j}$ is specified by the solution of the LP formulation. In contrast, the PRH heuristic assigns u to $p_{i,j}$ with a probability of $\mathcal{NR}_{u,i,j}$. The other steps are identical across the two techniques. Hence, the time complexities of the two techniques are also identical.

6 Results

In this section, we present the results obtained by running our formulation on four benchmarks namely, i) mp3 audio encoder ii) mp3 audio decoder iii) H.263 video encoder, and iv) 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 1. The communication trace graphs for the benchmarks were obtained from the work presented by Hu et. al [17]. Video processing requires more bandwidth than audio processing. Moreover, since the encoding process requires more processing than the decoding process, the H.263 encoder has a much higher bandwidth requirement, compared to H.263 decoder. Similarly, mp3 encoder has much higher bandwidth requirement compared to mp3 decoder. Among the four traces, mp3 decoder has the minimum bandwidth requirement, H.263 decoder and mp3 encoder have comparable bandwidth requirement, and H.263 encoder has the maximum bandwidth requirement.

In our experimental setup, we obtained results for different router architectures with 5 and 4 ports, respectively. We assumed an average power consumption of $0.6\mu W$ per mbps for each port of the router [4]. We utilized the Xpress-MP optimizer [18] to solve the MILP problems. The MILP formulation was run with a timeout at 43200 secs (12 hours). The best solution found after running the formulation for 12 hours was accepted. The sizes of the cluster were limited to 6 edges for the clustering based heuristic.

The results are summarized in Table 2. The power consumption, router requirement, and run time (normalized with respect to MILP solution) of the four techniques are shown in Figures 6, 7, and 8, respectively. The x-axis in the 3 plots refers to the serial number of the experiment. In the figures, results 1-10 and 11-20 are for 5-port and 4-port router architectures, respectively. Result number 15 in Figures 6, and 7 is empty since the MILP solution was unable to generate a solution in 12 hours. In Figure 8, the MILP runtime of result 15 is assumed to be 12 hours. As can be seen from Table 2 and the plots, the clustering based heuristic technique gave best results (within 11 % of the MILP solutions) in terms of power consumption, when compared with PDH and PRH techniques. The PRH technique gave slightly better results than PDH. In terms of the runtimes, the PDH technique was the fastest, followed by PRH, and finally the clustering based technique. The resource requirement of the three heuristic techniques were comparable, and were within 51 % of the MILP.

7 Conclusion

In this paper, we defined the application specific NoC synthesis problem and proposed linear programming based solutions. We presented an optimal formulation and several heuristic based techniques to overcome the long solu-

Graph	Graph ID	Nodes	Edges
mp3 encoder	G1	8	9
mp3 decoder	G2	5	3
263 encoder	G3	7	8
263 decoder	G4	9	8
263 enc 263 dec	G5	16	16
263 enc mp3 dec	G6	12	11
263 enc mp3 enc	G7	15	17
263 dec mp3 dec	G8	14	11
263 dec mp3 enc	G9	17	17
mp3 enc mp3 dec	G10	13	12

Table 1. Graph Characteristics

Technique	Deviation wrt MILP.				Avg run time (seconds)
	Power		Routers		
	Average	S.D	Average	S.D	
Clustering	1.11	0.19	1.51	0.34	171.1
PDH	1.58	0.55	1.51	0.49	1.90
PRH	1.47	0.55	1.51	0.58	1.92

Table 2. Summary of Results

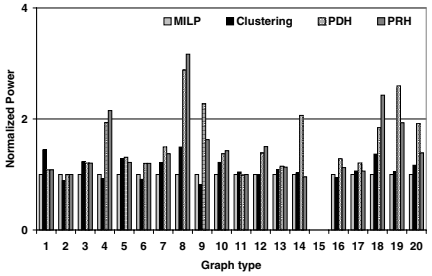

Figure 6. Comparison of power cons.

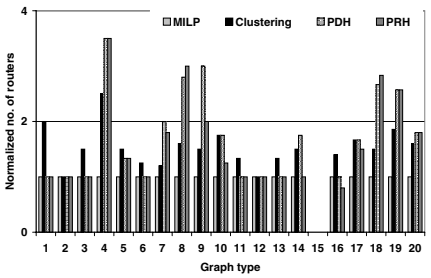

Figure 7. Comparison of router resources

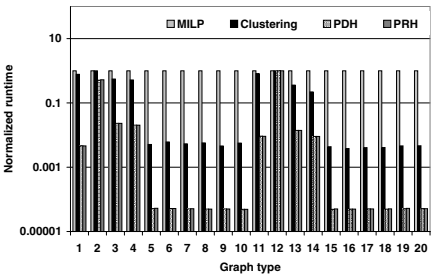

Figure 8. Comparison of run-time

tion times of the MILP formulation. We did extensive experimentation with four benchmarks namely, mp3 decoder, mp3 encoder, H.263 decoder, and H.263 encoder algorithms. The clustering based heuristic generated results whose power consumption was within 11 % of the MILP solutions and its average run time was 171.1 seconds. The average run time of the relaxation and rounding based techniques was less than 2 seconds, and the power consumption of their solutions was within 58 % of the MILP result.

Our formulations synthesize a NoC with homogeneous router architecture. The formulations can be easily extended to NoC synthesis with heterogeneous routers. The current formulations are aimed at minimizing dynamic power consumption, and do not address leakage power consumption. Future work will include developing dynamic heuristics for leakage power reduction. Further, the current formulations address the interconnection network design in isolation. Integration of low power design techniques for computation architecture with NoC synthesis have the potential to result in larger power savings, and would also be explored.

Acknowledgement : We would like to thank Dash Optimization [18] for their donation of the XpressMP solver.

References

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

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

[3] W.J. Dally and C.L. Seitz. "Deadlock-free message routing in multiprocessor interconnection networks". In *"IEEE Transactions on Computers"*, volume C-36, pages pp 547–553, 1987.

[4] 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.

[5] 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.

[6] T. T. Ye, L. Benini and G. De Micheli. "Analysis of Power Consumption on Switch Fabrics in Network Routers". In *Proceedings of DAC*, 2002.

[7] 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.

[8] 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.

[9] J. Luo, L-S. Peh, and N. K. Jha. "Simultaneous Dynamic Voltage Scaling of Processors and Communication Links in Real-time Distributed Embedded Systems". In *DATE*, 2003.

[10] A. Jalabert, S. Murali, L. Benini, and G. De Micheli. "xpipesCompiler: A tool for instantiating application specific Networks on Chip". In *DATE*, 2004.

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

[12] 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.

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

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

[15] D'andrade R. "U-Statistic Hierarchical Clustering". *Psychometrica*, 4(58-67), 1978.

[16] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Verlag, 1988.

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

[18] . "www.dashoptimization.com" .