# Application Specific Network-on-Chip Design with Guaranteed Quality Approximation Algorithms

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

*Abstract*— **Network-on-Chip (NoC) architectures with optimized topologies have been shown to be superior to regular architectures (such as mesh) for application specific multiprocessor System-on-Chip (MPSoC) devices. The application specific NoC design problem takes as input the system-level floorplan of the computation architecture, characterized library of NoC components, and the communication performance requirements. The objective is to generate an optimized NoC topology, and routes for the communication traces on the architecture such that the performance requirements are satisfied and power consumption is minimized. The paper discusses a two stage automated approach consisting of i) core to router mapping, and ii) topology and route generation for design of custom NoC architectures. In particular it presents an optimal technique for core to router mapping (stage i), and a factor 2 approximation algorithm for custom topology generation (stage ii). The superior quality of the techniques is established by experimentation with benchmark applications, and comparisons with integer linear programming (ILP) formulations, and heuristic techniques.**

## I. Introduction

Network-on-Chip (NoC) has been proposed by academia and industry as a solution for the on-chip communication challenges for the future MPSoC architectures. A NoC is composed of routers that are inter-connected by physical links. Each computation or storage core interacts with the NoC through a resource to network interface. The NoC supports packet switching based asynchronous communication. NoC is inherently scalable and supports high bandwidth by enabling concurrent communication.

Application specific MPSoC architectures that are aimed at a narrow domain such as multimedia set-top boxes integrate numerous heterogeneous computing and storage cores. Each core implements a limited set of application functionality. Consequently, inter-core communication depicts well defined patterns as determined by the specific domain. For such designs, the application specific custom NoC architecture has been demonstrated to be superior to regular topologies in terms of power consumption and NoC resources [1] [2].

The paper focuses on automated design of application specific custom NoC architectures. The overall MPSoC design flow can be divided into two stages focusing on computation architecture, and NoC architecture, respectively. The output of the computation architecture design stage is a collection of processing and storage cores, and inter-core communication performance requirements for NoC. The objective of NoC design stage is to construct an optimized interconnection architecture such that the communication performance requirements are satisfied and the power consumption is minimized.

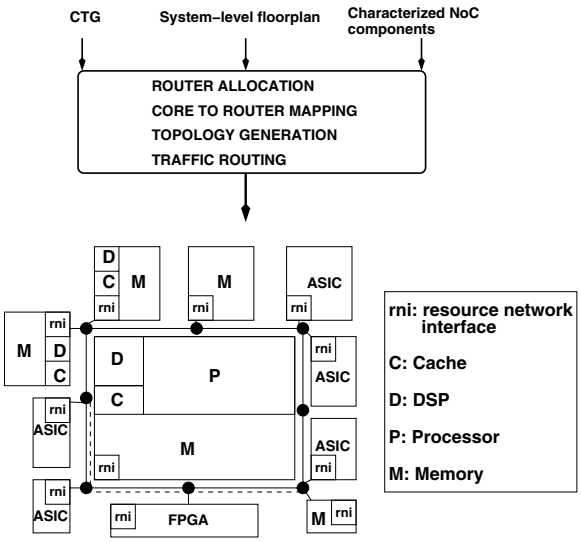


Fig. 1. Custom NoC design flow

It has been shown that the percentage of power consumption in NoC physical links increases from about 20% in 160 $nm$ technology [3] to upward of 30% in 65 $nm$ technology [2]. The power consumption in the physical links is directly dependent upon the length of the link and bandwidth of traffic flowing through the link. The length of the physical link is in turn governed by the system-level layout. The designer also specifies a maximum inter-router link length ($D_{max}$) for single clock cycle data transfer. Therefore, in order to account for link power consumption and to ensure that the link lengths are less than $D_{max}$, the design of the NoC architecture must consider the system-level layout. System-level floorplanning is a well researched topic, and existing techniques [4] can be utilized to generate the floorplans. Our automated NoC design techniques operate upon the system-level floorplan of the computation architecture.

### A. Custom NoC design flow

Figure 1 depicts the application specific NoC design flow. The inputs to the custom NoC design problem are: i) the system-level floorplan of the computation architecture, ii) communication bandwidth requirements, and iii) characterized component library of interconnection elements (namely, routers and physical links).

The NoC performance constraints are specified by a communication trace graph (CTG), where the nodes represent the

1-4244-0630-7/07/$20.00 ©2007 IEEE.

processing cores or memory elements, and the edges represent the communication traces between the cores. Each node in the CTG is specified by its physical dimensions, and each edge is specified by its bandwidth requirement. The cores in application specific MPSoC architectures for multimedia and network processing domains demonstrate well defined periodic inter-core communication characteristics and hence, can be easily modeled in the trace graph.

The output of the custom NoC design flow is a specification of the interconnection network topology, a mapping of computation architecture cores to the NoC routers, and routing of CTG traces on the topology such that the communication bandwidth requirements are satisfied, and power consumption is minimized.

In this paper, we address the NoC design problem by dividing it into two stages namely, core to router mapping, and routing and topology generation. We present a polynomial time optimal algorithm for mapping of cores to routers such that the power consumption is minimized. We also present a linear programming based approximation algorithm for routing and topology generation that minimizes the number of router resources in the topology subject to low power constraints. Our algorithm runs in polynomial time and guarantees that the number of routers in the topology is at most twice the optimal solution, and the power consumption is minimized.

### *B. Definition of the NoC design problem*

The NoC design problem was formally defined in [2]. We restate the problem for the sake of completeness.

" 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 core), and the directed 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 cycle.
- A router architecture characterized by:
  - a parameterizable value $\eta$, which denotes the number of ports of a router,
  - a value $\eta_{max}$ that denotes the maximum number of available ports in a router,
  - $\Omega$ denotes the peak input and output bandwidth that the router can support on any one port,
  - $\Psi_i$ that denotes the power consumed per unit bandwidth of traffic flowing in the input direction for any port of the router, and
  - $\Psi_o$ which denotes the power consumed per unit bandwidth of traffic flowing in the output direction for any port of the router.
- A physical link power model where $\Psi_l$ denotes the power consumed per unit bandwidth of traffic flowing through the link per unit length of the link.
- A system-level floorplan of the cores.
- A maximum inter-router distance $D_{max}$.

The objective of the custom NoC design problem is to construct:

- a NoC architecture $J(R, L, \mathcal{C})$, where $R$ denotes the set of routers utilized in the synthesized architecture, $L$ represents the set of links between two routers, and a many to one mapping function $\mathcal{C} : V \rightarrow R$ that denotes the mapping of a core to a router,
- a set $\mathcal{B}$ of ordered tuples of routers, where each $b_i\langle r_i, r_j, \ldots, r_k\rangle \in \mathcal{B}, r_i, \ldots, r_k \in R$ denotes a route for a trace $e(v_i, v_k) \in E$ ($\mathcal{C}(v_i) = r_i$, $\mathcal{C}(v_k) = r_k$), and

such that the NoC power consumption is minimized. "

## II. Previous work

Benini et al. [5] presented a survey of design techniques for application specific NoC architectures. Pinto et al. [6] proposed a technique for synthesis of point to point links that utilize at most two routers between source and sink. Thus, their problem formulation does not address routing. Jalabert et al. [1] proposed a custom NoC instantiation framework based on designer specified inputs. It does not synthesize the custom topology. Ogras et al. [7] proposed graph decomposition based heuristic techniques for application specific NoC architectures. Ogras et al. [8] also proposed heuristic incremental techniques that modified mesh based topologies via long link insertion. All papers cited above assume a constant predetermined link length, and hence do not incorporate system-level floorplanning to estimate their actual lengths. Srinivasan et al. [2] observed that the physical links are expected to consume upward of 30% of the NoC power consumption. They proposed integer linear programming (ILP) based techniques that incorporated system-level floorplanning to account for physical link power consumption. This paper accounts for link power consumption by taking the system-level floorplanning of the cores as input to the NoC design flow. To the best of our knowledge, this is the first attempt at designing approximation algorithms for NoC design The quality of the solutions produced by our technique is demonstrated by experimenting with several multimedia and network processing benchmarks, and comparisons with existing techniques.

The paper is organized as follows. In Section III we describe our technique for mapping of cores to the routers of the NoC. In Section IV, we present our technique that generates static routes for the traces, and generates the NoC topology. Section V presents the results, and finally, Section VI concludes the paper.

## III. Core to router mapping

Given a system-level floorplan, our technique assigns possible router locations to be the corners of the cores in the floorplan. Once the possible router locations are determined, our technique connects each core in the computation architecture to a unique router in the layout. Note that while several cores can be mapped to the same router, each core is uniquely mapped to a particular router. We assume that a core can be mapped to one of the four routers located at its corner. Thus, if $(x, y)$ denotes the lower left hand side corner of a node

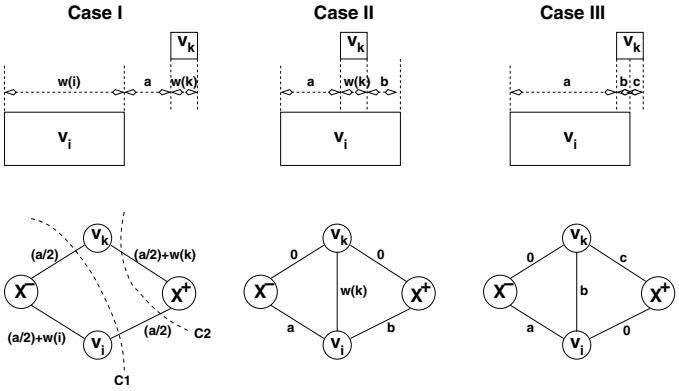


Fig. 2. Core alignments and flow graphs

$v$, the core is mapped to one of the routers located at $(x, y)$, $(x + \mathcal{W}_v, y)$, $(x, y + \mathcal{H}_v)$ or $(x + \mathcal{W}_v, y + \mathcal{H}_v)$, where $\mathcal{W}_v$ is the width of the core, and $\mathcal{H}_v$ is its height.

The objective of the core to router mapping is to minimize the power consumption. As the topology of the NoC is not defined at this stage, we abstract the power consumption as the power consumed by point to point physical links between two routers. For each core $i$, let $R_i$ denote the set of routers located at its four corners. Core $i$ is mapped to one of the routers in $R_i$. Let $X_{i,j}$ denote a (0,1) integer variable that is set to 1 if node $i$ is mapped to router $j \in R_i$, else 0. Let $X_{i,j,k,l}$ denote a (0,1) integer variable that is set to 1 if node $i$ is mapped to router $j$, and node $k$ is mapped to router $l$, else 0. We define these variables only when $(i, k) \in E$ or $(k, i) \in E$. The objective is to minimize the power consumption expressed as:

$$Minimize\ Z = \sum_{(i,k)\in E} \sum_{j\in R_i} \sum_{l\in R_k} \omega(i,k)\cdot\psi_l\cdot dist(j,l)\cdot X_{i,j,k,l} \tag{1}$$

where $dist(j, l)$ is the Manhattan distance between the two routers $j$ and $l$. In this section we prove that the core to router mapping problem is equivalent to max-flow min-cut problem, and therefore can be solved optimally in polynomial time.

### A. Equivalence to max-flow min-cut problem

The minimization goal can be split as the sum of two terms:

$$Z = \sum_{(i,k),j,l} \sigma(i,k)\cdot x(j,l)\cdot X_{i,j,k,l} + \sum_{(i,k),j,l} \sigma(i,k)\cdot y(j,l)\cdot X_{i,j,k,l} \tag{2}$$

where $\sigma(i, k) = \omega(i, k) \cdot \psi_l$, $x(j, l)$ is the x-offset between the two routers, and $y(j, l)$ is the y-offset between the two routers. We can consider the overall problem as a composition of two sub-problems that determine the $x$ and $y$ co-ordinate, respectively of the router to which the core is mapped.

Without loss of generality we first consider the sub-problem that determines the $x$ co-ordinates of the routers for all the cores. Thus, we wish to determine if the core should be mapped to a router in the $x^-$ (left hand of the core) or $x^+$ (right hand side of the core) location. Based on the relative locations of two communicating cores $v_i$ and $v_k$ on the floorplan we have three cases as shown in top row of Figure 2. For each case we construct a flow graph as shown in the lower row of the figure. In each graph we introduce two nodes $x^-$ and $x^+$ in addition to $v_i$ and $v_k$. We also add edges between the various nodes and annotate them with weights. The weight of an edge is derived from the various distances as specified in top row of the figure. A cut of each of the graphs that assigns $x^-$ and $x^+$ to different partitions denotes the mapping of the cores to the routers. The weight of the cut given by the summation of weights of edges that are cut denotes the x-offset between the routers to which the cores have been mapped. For example in Case I, the cut C1 denotes that $v_k$ is mapped to $x^-$ and $v_i$ is mapped to $x^+$. The weight of C1 given by $a$ denotes the x-offset between the $x^-$ router of $v_k$ and $x^+$ router of $v_i$. Similarly, the cut C2 denotes that $v_k$ and $v_i$ are both mapped to their respective $x^+$ router, the x-offset is $a + \mathcal{W}(k)$. The cut weight multiplied by the bandwidth of traffic between the two cores ($\omega(i, k)$) and link power model ($\psi_l$) denotes x-offset component of the power consumption due to the mapping (first term of equation 2). For a pair of communicating cores $(i, k)$ we can find the lowest power consumption router mapping by generating the minimum cost cut in the flow graph.

We now generalize the above construction to the entire CTG. We construct a flow graph $G(W, F)$ with $W = V \cup x^+ \cup x^-$ where $x^+$ and $x^-$ are additional nodes that represent the routers. For every trace $(i, k) \in CTG(V, E)$ we classify the trace in to one of the three categories based on the core locations, and introduce edges in flow graphs. The edge weights are given by the product of the communication bandwidth ($\sigma(i, k)$), link power model ($\psi_l$) and distance weights as described in the previous paragraph.

*Theorem 1:* A cut in Graph $G(W, F)$ that assigns $x^-$ and $x^+$ to different partitions represents a solution to the $x$ co-ordinate sub-problem.

*Proof:* Consider a core $v_i$. A cut either intersects the edge from $x^-$ to $v_i$, or the edge from $x^+$ to $v_i$, but not both. Moreover, a cut must intersect one of the two edges. If the edge from $i$ to $x^-$ is cut, it represents that core $v_i$ is mapped to router at $x^-$ off-set (and vice versa). The weight of the cut represents the power consumption contribution due to the mapping of the routers to the respective $x^+$ or $x^-$ routers. Thus, the cut captures both the mapping of the cores to the routers, as well as the power consumption incurred due to the assignments. Therefore, the cut represents a solution to the $x$ co-ordinate sub-problem. ■

From Theorem 1, it follows that if we could generate a cut of minimum cost, we have an optimal solution to the $x$ co-ordinate sub-problem. This is a polynomial time solvable max-flow-min-cut problem and we solve it by invoking the Push-Relabel algorithm [9] that has a complexity of $O(n^3)$ in the number of nodes of the graph. We can similarly solve the $y$ co-ordinate sub-problem, the utilize the two solutions to determine the unique router for mapping each core.

## IV. Routing and topology generation

The output of the first stage of our custom NoC design technique is a system-level layout with cores and routers,

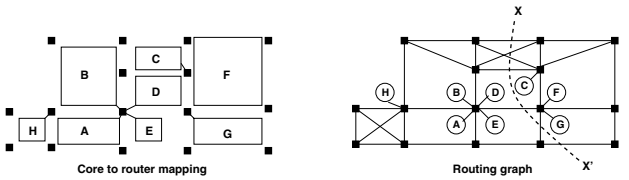


Fig. 3. Generating routing graph

and an assignment of the cores to specific routers (see left hand side of Figure 3). Thus, the source and sink routers for each trace are known. Our routing and topology generation technique addresses two problems namely, minimization of power consumption, and minimization of number of router resources in the network. We present our technique by first discussing a base case that minimizes the number of routers in the NoC, and build upon the base case to incorporate low power requirements.

### *A. Topology generation with least number of routers*

In this section, we present our technique to solve the problem of NoC topology generation such that the total number routers utilized in the topology is minimized. This problem is a node weighted generalized steiner forest problem, and is known to be NP hard [10]. We formulate the problem as an ILP. As mentioned before, it is known that integer relaxation of ILP formulations can be solved in polynomial time [9]. We prove that by utilizing iterative rounding on the integer relaxation of the ILP we can establish a 2-approximation on the quality of the solution in polynomial time. In other words the number of routers in the topology generated by our solution will be at most twice that of the optimal solution. In the following paragraphs, we first present a cut based ILP formulation for our problem. The cut based ILP has exponential constraints, and hence, is not suitable for practical applications. Therefore, we also present a equivalent flow based LP with polynomial number of constraints. We prove our 2-approximation bound by utilizing the LP of the cut based ILP, and iteratively solve the equivalent flow based LP to obtain the solution in polynomial time.

*1) Cut based ILP for minimizing routers:* Given the set of routers and core to router mappings, our technique initially determines the available paths from the source to the sink of each trace. This is done as follows. We construct a graph $G_r(V_r, E_r)$ called the routing graph, where $V_r$ is the set of available routers, and there exists an edge $e = \{r_i, r_j\}$ whenever the Manhattan distance between $r_i$ and $r_j$ is less than $D_{max}$. Figure 3 depicts a core to router mapping and the corresponding routing graph. In the figure, the black squares that denote the routers, are the nodes of the routing graph. The edges of the routing graph denote the possible physical links in the final NoC.

Let $\mathcal{C}$ denote a cut that divides the nodes of the routing graph $G_r$ into two subsets $S$ and $S'$. In Figure 3, the curve $X - X'$ represents a cut. For the cut, let $\delta(S)$ denote the set of edges that cross the cut. The edges intersected by cut $X - X'$ form the set of edges of $\delta$. For each cut $\mathcal{C}$ that divides the routers into sets $S$ and $S'$, we define a function called $F(\mathcal{S})$ that is set to 1 if there exists a trace such that its source lies in $S$ and sink lies in $S'$, or vice versa. Otherwise it is set to 0. Let $X_r$ denote a (0,1) variable that is set to 1 if router $r$ of the CTG is utilized in the NoC topology. Now, the LP can be formulated as follows:

$$Minimize\ Z = \sum_r X_r \ \ S.T$$

$$\forall \mathcal{C} \sum_{\forall e(r,s) \in \delta(\mathcal{S})} X_r \geq F(\mathcal{S})$$

The above constraint states that for a cut $\mathcal{C}$ that partitions the routers into sets $S$ and $S'$ such that $F(\mathcal{S}) = 1$, at least one router which is adjacent to the cut must be utilized in the final topology. Applying this constraint on all the cuts in the graph ensures that a route exists for all traces. Since the number of cuts in a graph is exponential, the cut based formulation defines a polytope with exponential constraints. We can reduce the number of constraints by utilizing an alternative flow based formulation. It is a well known fact that the cut based formulation and the flow based formulation are equivalent [11].

*2) Flow based ILP for minimizing routers:* The objective of the flow based formulation is same as that of the cut based formulation. Let $Y_{r,s,k}$ denote a (0,1) variable that is set to 1 if trace $k$ passes through the physical link $(r, s) \in L$. For each core $m$ in $V$, let $m\mathcal{M}r$ denote the relation that $m$ is mapped to router $r$. We have the following constraints :

- Any router that maps a core must be present in the network. Therefore, we have the following constraint.

$$\forall r, \exists m \in V : m\mathcal{M}r, X_r = 1$$

- A trace in the CTG always passes through the source and sink routers. For trace $t = (m, n)$, let $q$ denote the router that maps $m$, and $r$ denote the router that maps $n$. Since the trace is routed through the two routers, one physical link connected to the respective routers must be present in the topology. Hence, the following equalities must hold.

$$\sum_{q:(q,s)\in L} Y_{q,s,t} = 1 \quad \sum_{q:(s,q)\in L} Y_{s,q,t} = 0$$

The first constraint ensures that there is exactly one link through which the trace $t$ leaves router $q$. The second constraint ensures that the trace does not enter router $q$ from any other router. Therefore, the constraint ensures that there are no cycles in the traffic routes.
Similar constraints for the router mapping the sink are as follows.

$$\sum_{r:(s,r)\in L} Y_{s,r,t} = 1 \quad \sum_{r:(r,s)\in L} Y_{r,s,t} = 0$$

- For each trace, the flow due to the trace must be conserved in all routers except the routers that map the source and sink.

$$\forall t \in E \sum_{q:qr\in G_r} Y_{q,r,t} = \sum_{r:rs\in G_r} Y_{r,s,t}$$

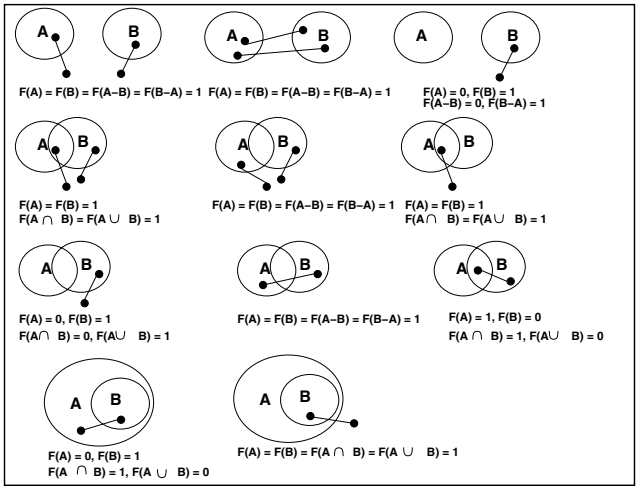


Fig. 4. Proof of supermodularity

- If there exists a flow through a router, that router is utilized in the topology.

$$\forall r, \forall k, X_r \geq \sum_{s:qr \in L} Y_{q,r,k}$$

The flow-based ILP only has $O(|E||L|)$ constraints. Therefore, the integer relaxation of the flow based ILP is suitable for direct solution by solvers.

*3) Algorithm and proof of 2-approximation:* In this section, we present our technique for obtaining integer solution from the LP relaxation, and prove that our technique utilizes at most twice the number of routers compared to the optimal solution.

*Definition 1:* A function $F : 2^R \rightarrow \mathbf{Z}$ is said to be weakly supermodular [10] if $F(R) = 0$, and at least one of the following conditions is true for any two sets $A, B \in R$ :

- $F(A) + F(B) \leq F(A - B) + F(B - A)$
- $F(A) + F(B) \leq F(A \cap B) + F(B \cup A)$

*Lemma 1:* The function $F$ defined in the cut based ILP is weakly supermodular.

*Proof:* As all traces are contained in the $G_r$, $F(R) = 0$. In order to prove the second part of the theorem, we enumerate all cases, and show that the supermodularity condition holds. The different cases are shown in Figure 4. The ovals A, B denote the two sets, the black circles denote two cores which may either lie inside or outside the sets, and the line joining the cores denotes the communication trace. Each case in the figure is annotated with one of the inequalities that satisfies the supermodularity conditions. ■

*Fact 1:* For any weakly supermodular function $F$, any extreme point solution $x$ to the LP must pick some router to the extent of at least a half. In other words, $X_r \geq \frac{1}{2}$ for at least one router $r$ [10].

Our algorithm exploits Lemma 1 and Fact 1, and utilizes an iterative rounding procedure on the integer relaxation solutions of the flow based ILP to generate the topology. The algorithm is presented below.

Begin algorithm **LP_round**
Initialize $H \leftarrow \phi : F' \leftarrow F$
While $F' \neq 0$ do
Solve LP to obtain solution $\mathbf{X}$.
For each variable $x \in \mathbf{X}$ do
If $(x \geq \frac{1}{2})$ then
round $x$ to 1
$H = H \cup x$
end-If
End for
Update $\forall S \subseteq V, F'(S) = F(S) - |\delta_H(S)|$
End While
Output H
End algorithm **LP_round**

*Theorem 2:* Algorithm LP_round achieves an approximation ratio of 2 for the routing problem.

*Proof:* Jain [10] gave an LP formulation for the edge weighted steiner tree problem, and utilized an iterative rounding procedure that results in factor 2 approximation for the problem. Our problem is an instance of the node weighted steiner tree. Consider an optimal solution $(X, Y)$ to our LP. First, note that $Y$ alone is feasible for the LP of Jain, because the constraints that do not involve $X$ variables in our LP are the same as the ones of Jain's LP: they require that a certain flow be supported (or that certain cuts be crossed) by the selected edges. Thus Jain's result implies that in an optimal solution to the LP there exists a triple $(p, q, t)$ such that $Y_{p,q,t} \geq 1/2$. Now for this particular $p$, we have the constraint $X_p \geq \sum_{\forall q,\forall t} Y_{p,q,t} + \sum_{\forall o,\forall t} Y_{o,p,t}$, and so $X_p \geq 1/2$. That is, in any optimal solution to our LP, there exists an $p$ with $X_p \geq 1/2$, and so in the for loop that the algorithm exercises in each iterative rounding phase, at least one more variable is fixed to 1. By Lemma 1, and Fact 1, our LP in the next phase satisfies all the requirements for Jain's theorem, and so the algorithm proceeds until a feasible integral solution is found.

Now we note that every time any variable $X_p$ was fixed to 1, this was either as part of an optimal LP solution, or because its fractional value was at least $1/2$. To prove the approximation bound, we use induction on the number of iterations (phases). In the base case, the algorithm only rounds up edges with fractional value at least $1/2$ and so the cost of the integral solution is at most twice the LP cost which is no more than the optimum. Suppose now for the induction step that $X$ is the LP solution found in the first iteration. Let $W$ be the set of vertices (routers) picked by rounding in the first iteration. By Fact 1, $W$ is nonempty. Let $X'$ be the vector we get from $X$ when we set all its components that are less than $1/2$ to zero. Since the rounding was applied only to the variables with values at least $1/2$, $cost(W) \leq 2cost(X')$.

Let $W'$ be the set of vertices picked in the later phases to satisfy the remaining requirements. The crucial observation is that $cost(W') \leq 2cost(X - X')$. This follows because $X - X'$ satisfies all the constraints that are not satisfied by $W$, and so is a feasible solution to the residual LP instance. By the induction hypothesis, the cost of the set of vertices picked in subsequent phases is at most twice the optimal residual LP solution $OPT'$, that is, $cost(W') \leq 2OPT' \leq 2cost(X - X')$. Now since $W + W'$ satisfies all the original constraints, we have cost$(W + W')$ $\leq$ cost$(W)$+cost$(W')$ $\leq$ 2cost$(X')$ +

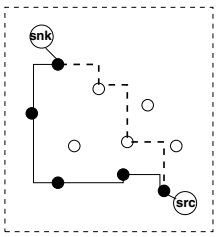


Fig. 5. Minimum power versus minimum routers

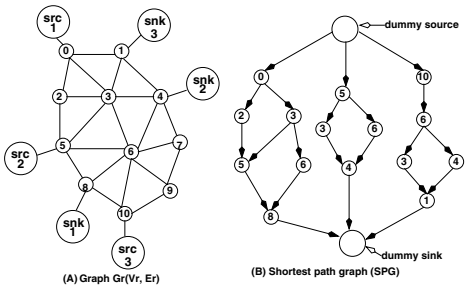


Fig. 6. Construction of SPG

$2\text{cost}(X - X') \leq 2\text{cost}(X)$. Comparing the leftmost side to the rightmost, we see that the total cost of vertices selected by the algorithm is at most twice the LP relaxation cost, and so at most twice the optimum. ■

### B. NoC topologies with minimum power consumption

For a trace $tr$, we define a shortest path route to be one that consumes minimum power. The total power consumption for the route is given by the power consumption due to the routers as well as the physical links. The technique presented in the previous section does not ensure shortest path routes for the traces. Therefore, some traces may be routed by longer paths, resulting in higher power consumption. For example, consider the route for trace $tr$ in Figure 5. In the figure, the darkened circles represent routers that are already present in the topology, and the empty circles are not present in the topology. A traffic route for $tr$ that consumes minimum number of routers is shown by heavy lines in the figure. Clearly, this is not the shortest path for the trace. The shortest path shown by dashed lines in the figure, consumes two extra routers. If the design objective is to minimize power consumption, the shortest path should be chosen at the expense of extra routers.

*1) Determination of shortest paths:* In order to determine the shortest paths, we again consider the routing graph. We associate a weight with each edge, which intuitively gives us the power consumption if the physical link corresponding to the edge is utilized in the final topology. For each edge $e = \{r_i, r_j\}$, $\rho_e$ denotes its edge weight, and is given by

$$\rho_e = \psi_i + \psi_o + l_e \times \psi_l$$

where $l_e$ denotes the Manhattan distance between $r_i$ and $r_j$. The edge weights are assigned such that they capture the link as well as the router power consumption.

Now, for each trace, we invoke Dijkstra's shortest path algorithm to determine the available shortest paths. Note that there can be multiple shortest paths from a source to a sink. The output of the call to the algorithm is a collection of subgraphs, each subgraph denoting the shortest paths for a particular trace. Consider the routing graph $G_r$ depicted in Figure 6(A). In the example, we are required to route three traces, $(src_1, snk_1)$, $(src_2, snk_2)$, and $(src_3, snk_3)$. The corresponding shortest path graphs are depicted in part (B) of the figure. Each node has a unique ID expressed as a double $(r, tr)$ where $r$ denotes the router in $V_r$, and $tr$ denotes the trace to which the node belongs. For example, router $(2, 1)$ denotes router with ID 2 in $V_r$, and belonging to the graph corresponding to trace 1. We denote this graph by *shortest path graph* or SPG.

We define $I(r, tr) = 1$ if node $(r, tr)$ is present in the solution obtained by invoking our routing and topology generation technique on the SPG. Otherwise, $I(r, tr) = 0$. We now define a mapping function, $I(r \times tr) \rightarrow X_r$ as follows.

$$X_r = \begin{cases} 1, \; if \; \forall tr \in E, \; \exists I(r, tr) = 1 \\ 0, \; otherwise \end{cases}$$

$X_r$ is set to 1 if any of the traces utilizes router $r$. Therefore, $X_r$ denotes the number of routers in the topology. We obtain our topology with minimum routers and shortest paths by invoking our linear programming based technique on the SPG, with an objective of minimizing the the sum of $X_r$. The cut based formulation is formulated as

$$Minimize \sum_r X_r \;\; S.T$$

$$\forall \mathcal{C} \sum_{\forall e((r,tr),(s,tr)) \in \delta(\mathcal{S})} X_r \geq F(\mathcal{S})$$

As before, we formulate an equivalent flow based formulation for the problem, and solve it in polynomial time with a 2-approximation guarantee.

### C. Post processing steps

At the end of the mapping and routing stages, the architecture may have routers that are placed very close to each other. In order to eliminate redundant routers, we merge pairs of routers that are less than distance $D_{max}$ apart. Our merging technique checks all pairs of available routers. Two routers are candidates for merging if the distance between them is less than $D_{max}$, merging the two routers does not violate the $D_{max}$ constraint for any other router in the topology, and the overall power consumption as a result of merging does not increase. The merging step is continued as long as there are pairs of routers that satisfy the above criteria.

Finally, our technique invokes post processing steps for bandwidth satisfaction, and deadlock avoidance. The width and number of physical links between two routers are increased such that they can support the required bandwidth. Potential deadlocks can be detected at design time by constructing the (virtual) channel dependency graph (CDG) and checking for cycles [12]. The cycles in the CDG can be removed by introducing additional virtual channels at selective router locations.

## V. Results

In this section, we present the results obtained by execution of our technique on several multimedia and network processing benchmark applications. We compare the results generated our technique (henceforth called guaranteed quality algorithms (GQA)) with an optimal ILP based technique [2], and a recursive partitioning based heuristic [13] called ANOC that addresses the same problem. The benchmarks included

| Graph | Graph ID | Nodes | Edges |
|---|---|---|---|
| JPEG Encoder | G1 | 8 | 21 |
| MPEG-4 decoder | G2 | 12 | 13 |
| MWD | G3 | 12 | 13 |
| VOPD | G4 | 12 | 13 |
| Set-top Box | G5 | 25 | 27 |
| AH Auth-IPv4 | G6 | 9 | 8 |
| Diffserv-IPv4 | G7 | 11 | 10 |
| NP1 | G8 | 15 | 22 |
| NP2 | G9 | 17 | 24 |
| NP3 | G10 | 24 | 42 |

Fig. 7. Benchmarks

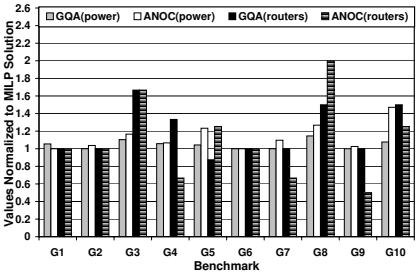


Fig. 8. Power and router comparisons for Case-1

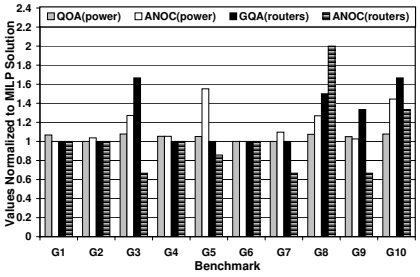


Fig. 9. Power and router comparisons for Case-2

5 multimedia applications (benchmarks G1 through G5) and 5 network processing applications (benchmarks G6 through G10). The sizes of the benchmarks ranged from 8 nodes and 21 edges to 24 nodes and 42 edges. The benchmarks were obtained from the work presented by Hu et al. [14], Pasricha et al. [15], and Ramamurthi et al. [16].

We present results for two cases: 1) NoC design subject to a maximum link length ($D_{max}$) constraint of $6mm$ henceforth called Case-1, and 2) NoC design without link length constraints, henceforth called Case-2. In accordance with the proof in Section III, the core to router mapping stage produced optimal solutions. The solutions generated by the routing and topology generation stage also converged to optimal solutions. Figures 8 and 9 present the comparisons of power and router resource consumption between our technique and the existing techniques. The bars in the figures are normalized to the optimal solution generated by the ILP based technique. In the figures, the first bar denotes the power consumption of our technique, the second bar denotes the power consumption of ANOC, the third bar denotes the router resource consumption of our technique, and the fourth bar denotes the router resource consumption of ANOC. Since the ILP based technique optimizes power consumption in isolation, our technique consumed less routers than the ILP for some benchmarks.

For Case-1 (Case-2), our technique consumed 1.04(1.04) times the power consumption of the optimal ILP based technique, and 0.93(0.9) times the power consumption of the heuristic technique. The corresponding values for router comparison were 1.12(1.21) and 1.21(1.31) respectively. The higher router consumption is due to the imposition of low power requirements that trades of the number of routers for lower power. While our technique generated results in less than a second, the ILP based technique took several hours to converge to the optimal solution. However, due to its lower complexity, the heuristic generated solutions in less than a second as well.

## VI. Conclusion

In this paper, we presented approximation algorithms for the design of custom NoC architectures. We presented polynomial time optimal and factor-2 approximation algorithm for the core to router mapping, and topology generation stages, respectively. We also presented graph transformations and heuristic techniques such that the NoC consumes minimum power, minimum router resources, and satisfies the architecture specific port and bandwidth constraints on the routers. We demonstrated the superior quality of the solutions generated by our technique by experimenting with an optimal ILP formulation, and an existing heuristic called ANOC.

## Acknowledgement

This work was partially supported by NSF grant (Career CCF-0546462 and CCF-0509540) and Consortium for Embedded Systems.

## References

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

[2] K. Srinivasan, K.S. Chatha, and G. Konjevod. Linear Programming based Techniques for Synthesis of Network-on-Chip Architectures. *IEEE Transactions on VLSI*, 14(4):407–420, 2006.

[3] S-J. Lee, S-J Song, K. Lee, J-H Woo, S-E Kim, B-G Nam, and H-J Yoo. An 800 mhz star connected on-chip network for application to systems on a chip. In *Proceedings of International Solid States Circuits Conference*, 2003.

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

[5] L. Benini. Application specific network-on-chip. In *Proceedings of DATE*, March 2006.

[6] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. Efficient synthesis of networks on chip. In *ICCD*, 2003.

[7] U. Ogras and R. Marculescu. Energy and Performance Driven NoC Communication Architecture Synthesis Using A Decomposition Approach. In *Proceedings of DATE*, 2005.

[8] U. Ogras and R. Marculescu. Application Specific Network-on-Chip Architecture Customization Via Long Range Link Insertion. In *Proceedings of ICCAD*, 2005.

[9] T.E. Cormen, C.E. Leiserson and R.L. Rivest. *Introduction To Algorithms*. McGraw-Hill, 1989.

[10] K. Jain . A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem . *Combinatorica*, 1:39–60, 2001.

[11] Pulleyblank . Polyhedral Combinatorics . *Handbooks on OR and MS*, 1:371–446.

[12] W. J. Dally and B. Towles. Route packet, not wires: On-chip interconnection networks. In *Proceedings of DAC*, June 2002.

[13] K. Srinivasan, and K. S. Chatha. A Low Complexity Heuristic for Design of Custom Network-on-Chip Architectures. In *Proceedings of DATE*, Munich, Germany, March 2006.

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

[15] S. Pasricha, N. Dutt, E. Bozorgzadeh, and M. Ben-Romdhane. FABSYN: Floorplan-Aware Bus Architecture Synthesis. *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, 14:241–253, 2006.

[16] V. Ramamurthi, Jason McCollum, Christopher Ostler, and K. S. Chatha. System-level Methodology for Programming CMP based Multi-threaded Network Processor Architectures. In *Proceedings of ISVLSI*, Tampa, Florida, USA, May 2005.