

# Key Research Problems in NoC Design: A Holistic Perspective

Umit Y. Ogras, Jingcao Hu, Radu Marculescu  
Department of Electrical and Computer Engineering  
Carnegie Mellon University  
Pittsburgh, PA 15213-3890, USA  
{uogras,jingcao,radum}@ece.cmu.edu

## ABSTRACT

Networks-on-Chip (NoCs) have been recently proposed as a promising solution to complex on-chip communication problems. The lack of an unified representation of applications and architectures makes NoC problem formulation and classification both difficult and obscure. To remedy this situation, we provide a general description for NoC architectures and applications and then enumerate several outstanding research problems (denoted by *PI-P8*) organized under three topics: communication infrastructure synthesis, communication paradigm selection, and application mapping optimization. Far from being exhaustive, the discussed problems are deemed essential for future NoC research.

## Categories and Subject Descriptors

J.6 [Computer Applications]: Computer-Aided Design – computer-aided design (CAD).

## General Terms

Algorithms, Performance, Design

## Keywords

Systems-on-Chip, Multi-processor systems, Networks-on-Chip

## 1. INTRODUCTION

Emerging *Network-on-Chip* (NoC) designs consist of a number of interconnected heterogeneous devices (*e.g.* general or special purpose processors, embedded memories, application specific components, mixed-signal I/O cores) where communication is achieved by sending packets over a scalable interconnection network.

The design of NoCs trades-off several important choices, such as topology selection, routing strategy selection and application mapping to network nodes. Developing a design methodology for NoC-based communication poses novel and exciting challenges to the EDA community. While a handful of design problems have been recently addressed by several researchers [1-5], a formal presentation of the major research themes is still missing. Having such a unifying formalism available can not only catalyze the research towards improving the already existing solutions, but also inspire new solutions to many outstanding research problems in NoC design.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CODES+ISSS'05, Sept. 19-21, 2005, Jersey City, New Jersey, USA.

Copyright 2005 ACM 1-59593-161-9/05/0009...\$5.00.

Rather than simply surveying prior work, the objective of this paper is to debunk some fundamental issues in NoC design and, more importantly, provide a roadmap for future research. Due to space limitations, this paper focuses primarily on the design methodologies aimed at the architectural-level of abstraction, only touching upon tightly coupled physical-level issues in NoC design. This way, this paper creates the basis for follow-up work that could target precisely these other levels of abstraction to complete the picture of NoC design.

To this end, we formally define a 3D design space that involves issues related to communication infrastructure synthesis, communication paradigm selection and application mapping optimization, then deal explicitly with each and every issue belonging to this problem space. In each case, we discuss motivation, problem formulation, proposed solutions and open research problems.

The paper is organized as follows: First, we introduce a novel formalism for application and architecture description. Then, in sections 3, 4 and 5, we discuss several outstanding research issues and suggest some open problems that deserve further consideration. Finally, we analyze the interaction among these research ideas and summarize our main contribution.

## 2. APPLICATION AND ARCHITECTURE DESCRIPTION

In this section, we propose an unified representation for NoC applications and architectures. At different stages in the design process (*e.g.* whether or not the application has been mapped, scheduled, *etc.*), the target application can be specified in one of the following two representations:

**Definition 1** A *Communication Task Graph* (CTG)  $G' = G'(T, D)$  is a directed acyclic graph, where each vertex represents a computational module in the application referred to as task  $t_i \in T$ . Each task  $t_i$  is annotated with relevant information, such as execution time on each type of Processing Element (PE) in the network, task  $i$  energy consumption ( $e_i^j$ ) when executed on the  $j$ -th PE, individual task deadlines ( $dl(t_i)$ ), periodicity of the task graphs, *etc.* Each directed arc  $d_{i,j} \in D$  between tasks  $t_i$  and  $t_j$  characterizes either data or control dependencies. Each  $d_{i,j}$  has associated a value  $v(d_{i,j})$ , which stands for the communication volume (*bits*) exchanged between tasks  $t_i$  and  $t_j$ .

**Definition 2** An *Application Characterization Graph* (APCG)  $G = G(C, A)$  is a directed graph, where each vertex  $c_i \in C$  represents a selected IP/core, and each directed arc  $a_{i,j}$  characterizes the communication process from core  $c_i$  to core  $c_j$ . Each  $a_{i,j}$  can be tagged with application-specific information (*e.g.* communication volume, communication rate, *etc.*) and specific design constraints (*e.g.* communication bandwidth, latency requirements, *etc.*). Also, the size/shape of cores  $c_i \in C$  is assumed to be known.

Compared to the CTG (Definition 1) which models applications at task- and communication transaction level, the APCG models the application at a coarser level, namely at IP/core level. Intuitively, the APCG can be derived from the CTG by binding and scheduling the tasks in the CTG onto different IPs.

Based on these background definitions, we can now introduce a novel description for NoC architectures which is one of the contributions of this paper.

**Definition 3** Different NoC architectures can be uniquely described by the triple  $\aleph(A(R, Ch), \mathfrak{R}, \Omega(C))$ , where the components have the following meanings:

- The directed graph  $A(R, Ch)$  describes the *communication infrastructure*; the routers ( $R$ ) and the channels ( $Ch$ ) in the network have attributes such as,
  - $\forall(ch) \in Ch, W(ch)$  gives the *width* of the network channels
  - $\forall r \in R, l(d, r)$  gives the input buffer size (*depth*) for the communication port  $d$  at router  $r \in R$ .
  - $\forall r \in R, P(r)$  specifies the position of the router  $r$ .
- $\mathfrak{R}(RD(r, s, d, \rho(n)), Sw)$  describes the *communication paradigm* adopted in the network.  $RD(r, s, d, \rho(n))$ ,  $s, d, r \in R, n \subseteq R$ , defines the routing policy at router  $r$  for all packets with source  $s$  and destination  $d$ . In this function,  $\rho(n)$  denotes the utilization of the neighboring routers, which can be used by an adaptive algorithm. Finally,  $Sw$  specifies the packet switching technique implemented in the network.
- $\Omega: C \rightarrow R$  maps each core  $c_i \in C$  to a router. For direct topologies, each router is connected to a core, while in indirect topologies some routers are connected only to other routers.

We can regard the architectural choices in the design of NoCs as representing a 3D design space, where each component of the triple  $\aleph(A(R, Ch), \mathfrak{R}, \Omega(C))$  defines a separate dimension. In the design automation community, design space exploration along each dimension has been performed to some extent without explicitly considering such a formalism. We present next a few approaches proposed to date and a number of open problems which are crucial for advancing the NoC research.

### 3. FIRST DIMENSION IN NOC DESIGN: COMMUNICATION INFRASTRUCTURE SYNTHESIS

The infrastructure synthesis aims at determining the communication architecture  $A(R, Ch)$ , given a particular application characterized by  $G = G(C, A)$  or  $G' = G'(T, D)$  (definitions 1 and 2). This design problem can be divided into four separate subproblems ( $P1-P4$ ) which are addressed next.

#### 3.1 P1: The Topology Synthesis Problem

##### 3.1.1 Motivation

The ability of the network to efficiently disseminate information depends largely on the underlying topology. Besides having a paramount effect on the network latency, throughput, area, fault-tolerance and power consumption, the topology plays an important role in designing the routing strategy and mapping the cores to the network nodes [6-15].

##### 3.1.2 Problem Formulation

**Given** an application graph  $G$  (or  $G'$ ), **find** the topology captured by  $A(R, Ch)$  which optimizes  $O(A, G)$ , subject to the constraints specified by  $Const(A, G)$ .

In this formulation,  $O(A, G)$  can be a metric for performance, area or reliability of the network, while the constraints  $Const(A, G)$  may be given by the amount of needed resources (e.g. area, wiring, etc.), bandwidth, and latency requirements imposed by the application. If the floorplan, channel lengths and widths are all known, more complex objective functions and/or constraints (e.g. power consumption, network latency, etc.) can also be utilized.

##### 3.1.3 Proposed Approaches

The simplicity of grid-like structures, as opposed to the complexity of custom topologies, inspired many design approaches where a mesh network is chosen a priori. Once the topology is fixed, the design problem reduces to application mapping to the given topology, while exploring different routing options [6-9] (also sections 4, 5). Such an approach may not satisfy the design objectives, or produce infeasible solutions, for some applications. Consequently, towards a better design space exploration, a richer set of standard network topologies is considered and the best one is selected [11,12].

The highest degree of flexibility is provided by customized topology synthesis which is desirable for several reasons. First, for application-specific NoCs, the detailed understanding of the communication workload can be exploited for optimization purposes [13-15]. Moreover, if the size/shape of the cores varies a lot, regular topologies waste area. Finally, the communication requirements of the components can vary widely. Designing the network to meet the requirements of highly communicating cores typically results in severe under utilization of other components, while designing it for average case may result in severe performance bottlenecks [10].

##### 3.1.4 Open Problems

Generally speaking, determining the optimal topology to implement any given application does *not* have a known theoretical solution. Although the synthesis of customized architectures is desirable for improved performance, power consumption and reduced area, altering the regular grid-like structure brings into the picture significant implementation issues, such as floorplanning, uneven wire lengths (hence, poorly controlled electrical parameters), etc. Consequently, ways to determine efficient topologies that trade-off high-level performance issues against detailed implementation constraints at micro- or nano-scale level need to be developed.

#### 3.2 P2: The Channel Width Problem

##### 3.2.1 Motivation

The width ( $W(ch)=W$ ) of the network channels comes into the picture in a number of places. First, the bandwidth of a network channel is given by

$$BW = f_{ch} \times W \quad [1]$$

where  $f_{ch}$  is the channel operating frequency. Moreover, in the absence of contention, increasing  $W$  reduces the message latency ( $L_0$ ). For instance, for a message consisting of  $SP$  bits (assuming wormhole routing and a flit size of  $W$ , see Section 4.2), when the source and destination nodes are  $H$  hops away,  $L_0$  is given by,

$$L_0 = H(t_r + t_s + t_L) + \max(t_s + t_L) \left\lceil \frac{SP}{W} \right\rceil \quad [2]$$

where  $t_r$ ,  $t_s$  and  $t_L$  are the times needed to make the routing decision, traverse the router and link, respectively [16]. While these relations suggest increasing  $W$ , we note that this can have side effects such as increasing the area (Section 3.3.1).

### 3.2.2 Problem Formulation

**Given** an application graph  $G$  (or  $G'$ ), and a communication architecture  $A(R,Ch)$ , **determine** the individual channel widths such that

- $\min(\text{network latency})$  and/or  $\max(\text{network throughput})$
- *subject to constraints*  $Const(A, G) \subset \{\text{area, total wire length, maximum wire length, power consumption, etc.}\}$ .

### 3.2.3 Proposed Approaches

Besides the aforementioned area effects, the choice of  $W$  has implications on wire sizing and spacing which determine the channel operating frequency. Hence,  $BW$  cannot be simply optimized by considering  $f_{ch}$  and  $W$  separately. Pileggi *et. al.* [25] discuss maximizing the channel throughput by controlling the number, size, and spacing of wires. Yet another problem is the issue of parallel vs. serial links in NoCs [38] under power/area constraints.

### 3.2.4 Open Problems

The selection of the channel width in NoC design has not been addressed to date. Dally *et. al* in [3] project a data width of 256 bits, while current NoC prototypes [26] use only 32-bit channel widths. Besides determining the optimal channel width for a given application, tools for analyzing various trade-offs involving the channel width, subject to wiring and area constraints, are needed for a fair comparison between different communication architectures.

## 3.3 P3: The Buffer Sizing Problem

### 3.3.1 Motivation

The input channel buffers at each router in the NoC have a serious impact on the overall area. For instance, by increasing the buffer size at each input channel from 2 to 3 words, the router area of a 4x4 NoC increases by 30% or more. Thus, the overall use of buffering resources has to be minimized to reduce the implementation overhead in NoCs.

At the same time, depending on the network workload, increasing the buffer size can reduce the network latency by orders of magnitude. However, due to the heterogeneity of traffic patterns in most application-specific NoCs, it makes sense to allocate more buffering resources *only* to the heavy loaded channels. Indeed, the values for average packet latency that can be obtained for the same total amount of buffering space are very different [24].

### 3.3.2 Problem Formulation

**Given:**

- Application communication characteristics defined by  $G$  (or  $G'$ ), such as packet injection rate at each IP, probability distribution of packet destinations, *etc.*
- Architecture specific parameters, such as routing function, router arbitration delay, link delay, *etc.* and the total available buffering space  $B$ .

**Determine:**

- Buffer size  $l(d, r)$  for *each* input channel at each router in the network which minimizes the average packet latency.

### 3.3.3 Proposed Approaches

The properties of on-chip buffers are studied in [22]. The authors report gate-level area estimates and analyze the performance of the network and buffers utilization across the network. Chandra *et al.* in [23] investigate the impact of FIFO

sizing on the interconnect throughput for single source, single sink interconnect scenarios.

An efficient algorithm for the buffer size allocation problem is proposed [24]. The approach assumes deterministic or oblivious routing, store-and-forward or cut-through switching schemes, and a Poisson distribution for all packets injected in the network. Under these assumptions, the authors derive the blocking rate of each individual channel and then add more buffering resources only to the highly utilized channels.

### 3.3.4 Open Problems

Although queuing theory can help achieving significant performance improvements through smart buffer allocation, many problems remain to be solved. The critical issue is the development of appropriate performance models that support

- *NoCs with wormhole routing.* Since the header flit and the payload flits are tightly coupled, deriving analytical performance models for NoCs under wormhole routing is a considerably more difficult problem.
- *Realistic traffic patterns.* Since real applications exhibit traffic patterns (*e.g.* [29]) which are very different compared to the Poisson model, the derivation of analytical models for performance evaluation becomes much more difficult.
- *NoCs with adaptive routing.* Adaptive routing is more difficult to analyze due to the increased traffic uncertainties.

## 3.4 P4: The Floorplanning Problem

### 3.4.1 Motivation

Due to their predictability, the communication architectures based on mesh or torus topologies alleviate the need to deal explicitly with issues at physical-level. However, if the size of the tiles in the network varies a lot, or arbitrary topologies are used, dealing with a floorplanning step becomes inevitable.

### 3.4.2 Problem Formulation

**Given**  $A(R,Ch)$ , a mapping  $\Omega(C)$  and the sizes of the cores, **find** a floorplan such that

- $\min(O(A, C, \Omega(C))) \subset \{\text{area, total wire length, etc.}\}$
- *subject to constraints*  $Const(A, G) \subset \{\text{area, total wire length, maximum wire length, etc.}\}$ .

### 3.4.3 Proposed Approaches & Open Problems

Floorplanning for NoCs needs to consider placement of routers and repeaters for latency insensitive operation [10,36]. Furthermore, the routability of the global network channel for maximum performance and well-controlled coupling effects needs to be taken into account. Floorplanning for regular topologies is addressed in [37]. Likewise, integrating the floorplanning with an application mapping algorithm is discussed in [11]. However, the floorplanning problem for arbitrary network topologies remains an open problem.

## 4. SECOND DIMENSION IN NOC DESIGN: COMMUNICATION PARADIGM SELECTION

The communication infrastructure alone does not capture the dynamic behavior of the network; this is determined by the communication traffic. The traffic flow is primarily governed by the communication paradigm, as detailed in this section, and the implementation of the target application, as discussed in Section 5 (Figure 1).

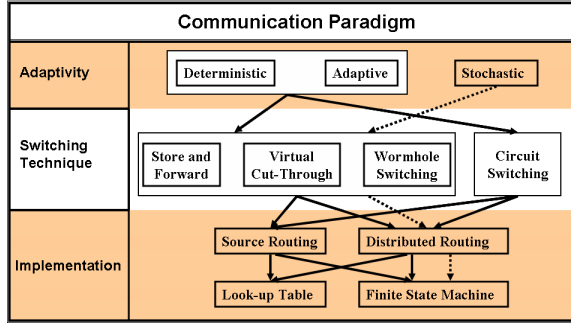


Figure 1. Classification of design choices for communication paradigm, similar to the one in [16].

## 4.1 P5: The Routing Problem

### 4.1.1 Motivation

An important problem in NoC design is deciding the type of routing; indeed, this greatly affects the network performance and power consumption [16-19]. Moreover, more complicated routing strategies result in larger design. Hence, this introduces interesting trade-offs between area and performance.

### 4.1.2 Problem Formulation

**Given** an application graph  $G$  (or  $G'$ ), a communication architecture  $A(R, Ch)$ , the source and destination routers, **find** a decision function at router  $r$ ,  $RD(r, s, d, \rho(n))$ , for selecting an output port to route the current packet(s), while achieving a certain objective function  $O(\mathfrak{R})$ .

$\rho(n)$  above represents the utilization of the channels connected to the routers in the set  $n \subseteq R$ , or the input buffers in those routers. Note that oblivious routing functions do not need this information, while adaptive algorithms usually check the congestion in the immediate neighbors. The routing decision can be solely based on local information, as well as on the global connectivity of the network specified by  $A(R, Ch)$ .

The objective function  $O(\mathfrak{R})$  can be selecting the minimal routing paths, avoiding congestion (while producing minimal paths or dropping minimal path requirement), avoiding deadlock, maintaining uniform power consumption across routers, improving fault-tolerance, etc.

### 4.1.3 Proposed Approaches & Open Problems

Implementation complexity and performance requirements are two major concerns in selecting the routing strategy. Compared to adaptive routing, deterministic routing requires less resources while guaranteeing an orderly packet arrival. On the other hand, adaptive routing provides better throughput and lower latency by allowing alternate paths based on the network congestion [18]. However, out-of-order message arrival remains an important problem associated with adaptive algorithms. Freedom from *deadlock* and *livelock* [16] are also crucial for NoCs, since deadlock (livelock) detection and recovery mechanisms are expensive and they may lead to unpredictable delays. Deterministic and partially adaptive algorithms based on the turn model [17], guarantee free deadlock and livelock operation, while fully adaptive strategies require extra precaution.

Deterministic routing is also more appropriate if the traffic generated by the application under consideration is predictable. Being so application dependent, the routing algorithm can indeed be customized (e.g. routing table allocation [6], traffic splitting [7], etc.) to match the application traffic pattern.

Stochastic routing for fault-tolerance in NoCs has been discussed in [20,21]. These studies are mostly experimental so theoretical results, performance models and prototypes are needed to justify the promise and energy overhead caused by redundant packet dissemination.

A power-aware, adaptive routing strategy that regulates the routing decisions to satisfy peak power constraints is proposed in [19]. Besides sharing the disadvantages of adaptive strategies, this approach does not address timing constraints, which are likely to coexist with power constraints. Hence, a power- and performance-aware technique is needed.

## 4.2 P6: The Switching Problem

### 4.2.1 Motivation

A problem related to routing is the switching technique used in the network. The switching technique determines when the routing decisions are made, how the switches inside the routers are set/reset, and how the packets are transferred along the switches [16]. Consequently, satisfying the constraints imposed by the application under different switching techniques results in trade-offs between implementation complexity (typically area) and performance.

### 4.2.2 Problem Formulation

**Given** an application graph  $G$  ( $G'$ ), and a communication architecture  $A(R, Ch)$ , **determine** the switching technique  $Sw \in \{store-and-forward, cut-through, wormhole, etc.\}$

- *subject to*  $Const(A, G) \subset \{end-to-end\ latency\ requirements, bandwidth\ requirements, area\ constraints, buffering\ and\ wiring\ resources, etc.\}$

### 4.2.3 Proposed Approaches & Open Problems

Among the commonly used switching techniques, wormhole switching seems to be the most promising one for typical NoC applications due to limited buffering resources and stringent latency requirements [3,6,7,31,32].

In data networks, wormhole routing is preferred to circuit switching due to the poor performance of the latter under dynamic traffic. However, for application-specific NoCs, this does *not* represent a major handicap. Moreover, guaranteed service operation, as required by some applications, is relatively easier to satisfy by using circuit switching [33,34] as opposed to wormhole routing [31,32]. Therefore, circuit switching is a promising alternative, despite its implementation complexity and static nature. It remains to be seen whether or not a particular switching technique, or a hybrid combination [35], is more advantageous.

## 5. THIRD DIMENSION IN NOC DESIGN: APPLICATION MAPPING OPTIMIZATION

The mapping problem aims at determining how to map an application onto the NoC platform, while certain metrics of interest (energy, performance, etc.) are optimized. Depending on the flexibility of the given NoC platform, the mapping problem may have different flavors as we discuss below.

### 5.1 P7: The Scheduling Problem

#### 5.1.1 Motivation

We use the term *hard NoC* to represent NoC platforms where *both* computation and communication have been pre-designed. Since such platforms offer no real flexibility for architectural

customization, the mapping issue reduces to solving the communication and task scheduling problems.

Although the scheduling problem is a traditional topic in computer science, most previous work neglects the inter-processor communication. Alternatively, it assumes a fixed delay proportional to the communication volume, without taking into consideration subtle effects (*e.g.* communication congestion) which change dynamically throughout tasks execution. Thus, the scheduling problem remains an important problem for NoC design.

### 5.1.2 Problem Formulation

A simple but important category of problems is off-line static scheduling. To this end, we first give the definition of *Architecture Characterization Graph* by extending Definition 3 to include the behavior of the computational nodes. Then, we formulate the problem of energy-aware scheduling for heterogeneous NoCs under deterministic routing.

**Definition 4** A NoC *Architecture Characterization Graph* (ACG) can be uniquely described by the 4-tuple  $\mathcal{N}(C, A(R, Ch), \mathfrak{R}, \Omega(C))$ . Here  $C$  represents the set of cores/PEs in the NoC, while  $A(R, Ch)$ ,  $\mathfrak{R}$  and  $\Omega(C)$  remain the same as in Definition 2.

Let  $e(r_{C_i, C_j})$  denote the energy required to send one bit from PE  $C_i$  to  $C_j$ . Using these definitions, the energy-aware scheduling problem for heterogeneous NoCs is formulated as:

**Given** a CTG ( $G'$ ) and an ACG, **find** a mapping  $M(\cdot)$  from the set of tasks ( $T$ ) to the set of PEs ( $C$ ), together with a starting time for each task and communication transaction, which minimizes.

$$\min \left\{ \text{Energy} = \sum_{\forall t_i} e_{M(t_i)}^i + \sum_{\forall d_{i,j}} v(d_{i,j}) \times e(r_{M(t_i), M(t_j)}) \right\} \quad [3]$$

such that

- All the control and data dependencies are satisfied.
- All tasks  $t_i$  for which individual deadlines  $dl(t_i)$  are specified finish execution before or at their respective deadlines.

### 5.1.3 Proposed Approaches

Architectural support for compile-time scheduling for on-chip communication is presented in [26]. This approach optimizes data transfers that can be determined at compile-time using scheduling, and provides a software-based dynamic routing.

An algorithm for the energy minimization problem is proposed in [27]. The algorithm first allocates more slack to those tasks which have a larger impact on energy consumption and performance. A level-based scheduling mechanism is then used to schedule the tasks and communication transactions in parallel. Finally, a search and repair procedure iteratively improves the solution by fixing possible deadline misses in the schedule generated by the first two steps.

### 5.1.4 Open Problems

The algorithm in [27] targets real-time or DSP applications where the worst-case task execution time and inter-task communication volume are pre-characterized. Thus, such applications can be modeled as CTGs which expose the inherent inter-node parallelism existing in the application. However, such a fully static scheduling, can *not* be directly applied to applications containing conditional branches [28]. When applications behavior can not be predicted at compile time, *on-line* scheduling approaches are usually needed. Significant work is needed to develop efficient performance- and energy-aware on-line scheduling algorithm for NoCs.

## 5.2 P8: The IP Mapping Problem

### 5.2.1 Motivation

Another important category of NoC platforms, called *firm NoCs* in this paper, are those where the communication architecture has been pre-designed, but the designer can still decide how to embed different IPs onto different tiles that act as placeholders in the architecture. More precisely, given an application described by a set of concurrent tasks, already bounded and scheduled onto a list of selected IPs, the problem is to determine how to topologically *map* the selected IPs onto the network, such that certain metrics of interest are optimized.

To optimally use a firm NoC platform, the task and communication scheduling/binding and the IP mapping should be ideally performed in parallel. However, such a task would be too complex in practice. Therefore, the task and communication scheduling/binding can be fixed *before* the IP mapping stage and then, the results after mapping can be used as feedback to further tune the scheduling and binding.

### 5.2.2 Problem Formulation

**Given** an APCG,  $G = G(C, A)$ , and a network architecture  $A(R, Ch)$ , **determine** a mapping  $\Omega : C \rightarrow R$  such that:

- $\min(O(A, G) \subseteq \{\text{energy, performance, thermal behavior, fault-tolerance etc.}\})$
- $\text{subject to } \text{Const}(A, G) \subseteq \{\text{area, bandwidth and/or latency requirements, etc.}\}$
- each IP/core  $c_i \in C$  is mapped to a router  $r \in R$ ,

### 5.2.3 Proposed Approaches

IP mapping for regular NoCs is addressed by Hu *et. al.* [6] where a branch and bound algorithm is proposed to map a given set of IPs onto a NoC architecture, while minimizing the total communication energy. Similarly, Murali *et. al.* [7] propose an efficient mapping algorithm for NoC architectures which supports traffic splitting. These techniques use the *average packet hop* value as a cost function by relating it to the communication energy consumption [6] or communication cost [7]. Multi-objective mapping to mesh-based NoC architectures is discussed in [9]. If the IPs have different sizes, the communication latency and power consumption (per data unit) between any two neighboring routers may differ significantly. To calculate the energy or latency, it is therefore necessary to embed the floorplanning right inside the mapping loop in order to get better results at the expense of increased complexity [11].

With the increasing power density and cooling costs, it is important to reduce, or even eliminate, the potential hotspots and obtain a thermally-balanced design. In [8], a genetic algorithm is proposed to design a thermal balanced design while minimizing the communication cost via placement.

### 5.2.4 Open Problems

Simulation is, in general, too costly to use in an optimization loop, so a key component to have in solving the IP mapping problem is a good analytical model for performance evaluation. Depending on the optimization objective, different analytical models may be needed. For instance, for communication energy minimization, an accurate energy model is essential. An energy model based on average packet hop is presented in [6], but other models are required if the underlying architecture (*e.g.* for irregular architectures) or the optimization objective changes (*e.g.* IP mapping to maximize performance). Nevertheless, there are no good analytical models readily available to use for performance-aware mapping since, compared to energy models, deriving a good performance model is much more difficult. The

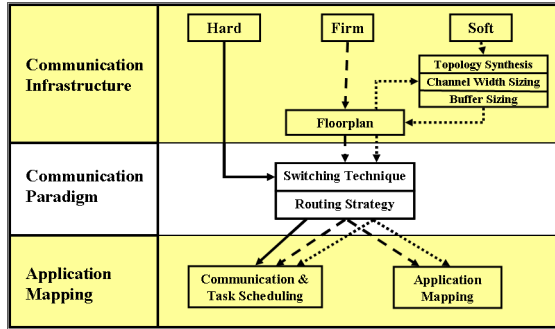


Figure 2. Illustration of design space for NoC architectures.

use of average packet hop is possible only if the network is *not* congested. Unfortunately, for on-chip applications, the communication architecture is mostly used in regimes closer to congestion. As such, the queuing delay of a packet (or flit) becomes dominant so the estimation method based on average packet hop is not appropriate. The development of such advanced models remains an open challenge.

## 6. INTERACTION AMONG THE RESEARCH PROBLEMS

Having discussed all individual issues, it makes sense now to analyze the interaction among these research problems. Such an interaction is summarized in Figure 2 using again the same 3D view on the design space. More precisely:

- Along the communication infrastructure dimension, there are three possible approaches. The first possibility is using *hard NoC* platforms, which have fixed architectures and both computation and communication components have been pre-designed. As a result, the design process essentially reduces to exploiting the communication paradigm ( $P5, P6$ ) and solving the communication and task scheduling problem ( $P8$ ).
- Next possibility is using *firm NoCs*, where the communication architecture is already pre-designed but the designer can choose how to embed individual IPs onto different tiles which act as placeholders in the architecture. The problems to be solved for customizing these platforms are floorplanning ( $P4$ ), communication paradigm ( $P5, P6$ ), application mapping ( $P7$ ).
- Finally, the last alternative of having *soft NoC* platforms offers the maximum flexibility for customization. The designer has the freedom of customizing topology ( $P1$ ), network channels ( $P2$ ) and buffer allocation ( $P3$ ), as well as addressing the problems ( $P4-P8$ ), similar to previous approaches.

## 7. CONCLUSION

In this paper, we have presented several major research challenges in the design of NoCs. Rather than simply enumerating relevant prior work, we have provided a discussion for each problem including motivation, formulation, and open research issues. The problems addressed in this work are focused at the architectural-level, while future work can cover the other levels of abstraction using a similar philosophy.

Far from being exhaustive, we plan to move the problems addressed in this work to a public repository where researchers can add new problems, solutions or software tools. Such a dynamic companion can transform the ideas of this paper into a valuable tool for NoC research.

**Acknowledgements:** This research is supported by Marco GSRC, NSF CCR-00-93104, and SRC 2004-HJ-1189.

## 8. REFERENCES

- [1] A. Jantsch, H. Tenhunen (Eds.). Networks-on-Chip. Kluwer, 2003.
- [2] S. Kumar, *et al.* Network on a chip: An architecture for billion transistor era. In Proc. IEEE NorChip Conf, 2000.
- [3] W. Dally, B. Towles. Route packets, not wires: On-chip interconnection networks. In Proc. of DAC, June 2001.
- [4] L. Benini, G. De Micheli. Networks on chips: A new SoC paradigm. IEEE Computer. 35(1), 2002.
- [5] J. Henkel, *et al.* On-chip networks: A scalable communication-centric embedded system design paradigm. In Proc. VLSI Design 2004.
- [6] J. Hu, R. Marculescu. Energy- and performance-aware mapping for regular NoC architectures. IEEE Trans. on CAD of Integrated Circuits and Systems, 24(4), April 2005.
- [7] S. Murali, G. De Micheli. Bandwidth-constrained mapping of cores onto NoC architectures. In Proc. DATE, 2004.
- [8] W. Hung, *et al.* Thermal-aware IP virtualization and placement for Networks-on-Chip architecture. In Proc. ICCD, 2004.
- [9] G. Ascia *et al.* Multi-objective mapping for mesh-based NoC architectures. In Proc. CODES, 2004
- [10] A. Jalabert, *et al.* xpipesCompiler: A tool for instantiating application specific Networks on Chip. In Proc. DATE, 2004.
- [11] S. Murali, G. De Micheli. SUNMAP: A tool for automatic topology selection and generation for NoCs. In Proc. DAC, 2004.
- [12] M. Kreutz, *et al.* Communication architectures for System-On-Chip. Symp. on Integrated Circuits and Systems Design, 2001.
- [13] A. Pinto, *et al.* Efficient synthesis of networks on chip. In Proc. ICCD, Oct. 2003.
- [14] K. Srinivasan, *et al.* Linear programming based techniques for synthesis of Network-on-Chip architectures. In Proc. ICCD, 2004.
- [15] U. Y. Ogras, R. Marculescu. Energy- and performance- driven customized architecture synthesis using a decomposition approach. In Proc. DATE, 2005.
- [16] J. Duato, *et al.* Interconnection Networks: An Engineering Approach. Morgan Kaufmann, 2002.
- [17] C. J. Glass, L. M. Ni. The turn model for adaptive routing. In Proc. ISCA, May 1992.
- [18] J. Hu, R. Marculescu. DyAD-Smart routing for Networks-on-Chip. In Proc. DAC, June 2004.
- [19] L. Shang *et al.* PowerHerd: Dynamically satisfying peak power constraints in interconnection networks. In Proc. Intl. Symp. on Supercomputing, June 2003.
- [20] T. Dumitras, R. Marculescu. On-Chip stochastic communication. In Proc. DATE, March 2003.
- [21] M. Pirretti *et al.* Fault tolerant algorithms for Network-On-Chip interconnect. In Proc. IEEE Symp. on VLSI, February 2004
- [22] I. Saastamoinen, *et al.* Buffer implementation for Proteo Network-on-Chip. In Proc. Intl. Symp. on Circuits and Systems, 2003.
- [23] V. Chandra, *et al.* An interconnect channel design for high performance integrated circuits. In Proc. DATE, Feb. 2004.
- [24] J. Hu, R. Marculescu. Application-specific buffer space allocation for Networks-on-Chip router design. In Proc. ICCAD, 2004.
- [25] T. Lin, L. T. Pileggi. Throughput-driven IC communication fabric synthesis. In Proc. ICCAD 2002.
- [26] J. Liang, *et al.* An architecture and compiler for scalable on-chip communication. IEEE Trans. on VLSI Systems, 12(7), July 2004.
- [27] J. Hu, R. Marculescu. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints. In Proc. DATE, 2004.
- [28] Y. Xie, W. Wolf. Allocation and scheduling of conditional task graph in hardware/software co-synthesis. In Proc. DATE 2001.
- [29] G. Varatkar, R. Marculescu. On-chip traffic modeling and synthesis for MPEG-2 video applications. IEEE Trans. on VLSI, 12(1), 2004.
- [30] <http://cares.icsl.ucla.edu/MediaBench/applications.html>
- [31] E. Bolotin, *et al.* QoS architecture and design process for Networks-on-Chip. Journal of Systems Arch., vol. 50, 2004.
- [32] T. Bjerregaard, J. Sparso. A Router architecture for connection-oriented service guarantees in the MANGO clockless Network-on-Chip. In Proc. DATE, 2005.
- [33] J. Dielissen, *et al.* Concepts and implementation of the Philips Network-on-Chip. IP-based SoC Design, Nov. 2003.
- [34] M. Millberg, *et al.* Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In Proc. DATE, 2004.
- [35] K. Lee, *et al.* A 51mW 1.6GHz on-chip network for low-power heterogeneous SoC platform. Intl. Solid-State Circuits Conf., 2004.
- [36] L. P. Carloni, *et al.* Theory of latency-insensitive design. IEEE Trans. on CAD of Integrated Circuits and Systems, 20(9), Sept. 2001.
- [37] T. T. Ye, G. De Micheli. Physical planning for multiprocessor networks and switch fabrics. In Proc. ASAP, 2003.
- [38] A. Morgenshtein, *et al.* Comparative analysis of serial vs. parallel links in networks on chip. Intl. Symp. on System-on-Chip 2004.