# A Methodology and an Open Software Infrastructure for Constraint-Driven Synthesis of On-Chip Communications

*Alessandro Pinto*
*Alberto L. Sangiovanni-Vincentelli*
*Luca Carloni*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-130
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-130.html

November 4, 2007

# A Methodology and an Open Software Infrastructure for Constraint-Driven Synthesis of On-Chip Communications

Alessandro Pinto, *Member, IEEE,* Luca P. Carloni, *Member, IEEE,* and Alberto L. Sangiovanni-Vincentelli, *Fellow, IEEE*

***Abstract*— An SoC is modeled as a set of pre-designed cores characterized by their area and interfaces and a set of end-to-end communication requirements on the channels connecting them. These requirements are the input specification to a synthesis process that produces an on-chip communication (OCC) by assembling components such as interfaces, routers, buses and links, from a target library. Models for functionality, cost, and performance of each element are captured in the library together with their composition rules. Our contributions is two pronged:**

1) **A methodology for the design of OCC based on:**
   - **a mathematical framework to model communication at different levels of abstraction from the point-to-point input specification to the library elements and the final implementation;**
   - **the formulation of an optimization problem and a heuristic synthesis algorithms to solve it: given a set of communication requirements on the throughput and latency of the channels it returns a communication implementation that satisfies them while minimizing power consumption.**
2) **A publicly available software package that includes alternative synthesis algorithms, various communication libraries, supporting tools for simulation and visualization, and a collection of SoC benchmarks.**

***Index Terms*— Communication synthesis, System-on-chip, Interconnect synthesis, Performance optimization.**

## I. Introduction

With the advances of IC technology, global interconnects have become the dominant factor in determining chip performance: they are not only becoming responsible for a larger fraction of the overall delay and power dissipation but exacerbate also design problems such as noise coupling, routing congestion, and, timing closure, thereby imposing severe limitations on design productivity [1]. Because of these characteristics, most VLSI circuits can be considered *distributed systems*, a fact that challenges traditional design methodologies and the electronic design automation tools that are based on them. Systems-on-Chip (SoCs) are typically designed by assembling intellectual property (IP) components from different vendors and/or different divisions of the same company in the attempt of reducing time-to-market by reusing pre-designed and pre-verified elements. However, since these components are designed independently, the assembly step is often a challenging problem that requires the design of communication interfaces to match different protocols and data parallelisms and the routing of global interconnect wires to meet the constraints imposed by the target clock period.

The Open Core Protocol (OCP) [2] tackles this problem by defining a standard open-domain interface with which IP cores should comply to allow fast integration using appropriate interconnect architectures. While there is no intrinsic limitation on the interconnect architecture for OCP, most designers rely on traditional bus architectures so that pre-designed components can be used. In this domain, proprietary protocols such as the ARM AMBA BUS and the IBM CoreConnect are popular among SoC designers making the adoption of a universal standard difficult at best.

We argued that SoCs are distributed systems. For this reason, bus architectures may not be always ideal; in fact, scalable, multi-hop, packet-switched Networks-on-Chip (NoCs) have been proposed in a set of seminal papers [3]–[5] as a solution for the integration of IP components as an interesting alternative. Borrowing from the communication networks literature, the components that can be combined to build an NoC can be heterogeneous including elements such as interfaces, routers, and links. Because of the many degrees of freedom in NoCs (such as choice, topology, and positions of the communication components, core interfaces, protocols) and composition rules, and of the many objectives that are of interest (such as performance, power consumption, reliability, and occupied area) to find an optimal or just a good solution is a very challenging proposition. Hence, the NoC design problem had been simplified by limiting the number and types of components considered, by focusing on a subset of the relevant objectives, by constraining NOC topology and components positions, and by dividing the optimization process in successive stages. Limiting the degrees of freedom has also the important side effect of reducing implementation and layout complexity. In NETCHIP [6] an *application-specific* NoC is obtained by mapping the application cores on standard topologies (e.g torus, mesh, hypercube) in an optimal way (SUNMAP [7]). In [8], Hu and Marculescu perform mapping and routing on the NoC with optimal energy and performance. Lahiri *et al.* use standard topologies consisting of collections of channels (point-to-point links or shared busses) interconnected by bridges [9]. Ogras *et al.* [10] proposed a perturbation method that starting from the mapping of an application on a standard topology, optimizes the initial performance and cost by inserting custom long links between routers. In [11] Murali *et al.* synthesize NoCs that, albeit being more general than the approaches that start from a regular topology, are still

A. Pinto and A. Sangiovanni-Vincentelli are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA ({apinto,alberto}@eecs.berkeley.edu). L.P. Carloni is with Department of Computer Science, Columbia University in the City of New York, New York, NY 10027, USA (luca@cs.columbia.edu).

constrained to be "two-level structures", where star topologies are interconnected by links chosen to to satisfy inter-cluster communication requirements.

Srinivasan *et al.* proposed to synthesize an application-specific NoC without assuming any pre-existing interconnection fabric [12]. The design flow is based on floorplan, generation of admissible router positions, and optimal routing. The synthesis problem is solved via integer linear programming (ILP). Due to the complexity of ILP, only few locations for the installation of routers are considered. This simplification still yields running time of the order of several hours even for relatively small instances. More recently, the same authors proposed an efficient approximation algorithm that guarantees the optimality of the solution within a certain bound from the global optimum [13]. This bound is strongly tied to the cost model. Specifically, the per-hop cost is lumped on the NoC links and is assumed to be linear in the bandwidth, while no constraints are imposed on the router size.

We showed that a rich set of interesting results for NoCs exist; however, few are the examples of practical applications of NoCs. In fact, the debate between those who favor standard bus architectures or variations thereof and those who advocate the adoption of NoC approaches ranging from constrained architectures to custom ones is vibrant. We do not take sides even though the NoC approach has undisputable fundamental merits that may make it successful in the long run. Instead, we propose a general methodology for the design of on-chip communication that can explore a large number of alternatives including as special cases NoCs, bus architectures and hybrid ones. Given the generality of our approach, it can be used to build a framework where different constrained solutions can be compared using a number of evaluation factors.

Our approach is based on the synthesis of optimal heterogeneous network topologies by assembling components from a fine-grained library without enforcing any constraint on the topology other than the ones formally captured in the library. In particular, the network that we obtain need not be direct and not even connected if these constraints are not captured in the composition rules of the communication components. The possible choice by the synthesis algorithms for a multistage network rather than a direct network is a consequence of the number of concurrent accesses to shared resource (e.g. DRAM controllers) and of the constraints on the amount of communication resources provided by the routers (i.e. number of input and output ports) and by the links (i.e. bandwidth capacity).

We borrow the general approach offered by the Platform-Based Design (PBD) methodology [14]. In PBD, the design specification and the implementation alternatives are kept separate. The methodology is recursive: the functional specification is implemented on a particular architecture through a series of refinement steps. At each step, which corresponds to a specific level of abstraction, the implementation alternatives are characterized by a set of components that can be instantiated, configured, and assembled according to specific rules, to derive a more complex structure. The set of components together with the rules that specify how to compose them define a *platform* on which design requirements

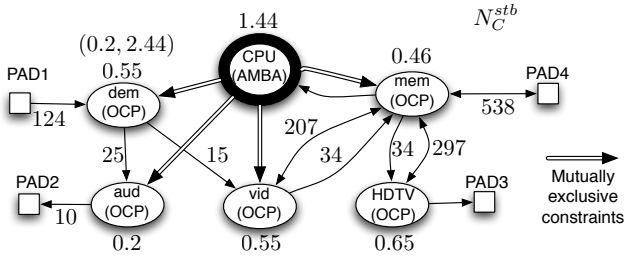


Fig. 1. The system-level specification of a simplified Set-Top Box. Each core in the specification is annotated with and area in $mm^2$ and each arrow is annotated with a bandwidth constraint in $MB/s$.

can be mapped to define a possible implementation. In this sense, a platform is a family of admissible solutions. The task of the synthesis process is then to select one out of this family (a *platform instance*) and a mapping of the requirements onto the components that satisfy the requirements and possibly optimize the objectives of the design. The implementation refines both requirements and platform instance and is defined at a lower level of abstraction.

In this process, it is essential to formalize how requirements are specified, how the library is described, and how the composition rules are defined and applied to generate the space of admissible solutions. The composition rules can be used to encode constraints related to the topology that the designer wishes to consider while the components present in the library determine which kind of "nodes" can be selected. To select a platform instance using an optimization algorithm we must associate to each library element (and to the hierarchical composition of two or more of them) a "characterization" in terms of cost, performance, power, and "type" (e.g., number of ports and interface type of a router) that allows us to evaluate metrics associated with the objectives and constraints of the design. In addition to the optimization framework, we also provide a heuristic "meta-algorithm" that can be refined into various ad-hoc algorithms depending on the constraints that the designer is most concerned with.

To illustrate our approach, consider, for instance, the simplified *Set-Top Box System* shown in Figure 1. This design will serve as an example throughout the paper. The SoC *specification* contains six IP cores that exchange messages through a dozen of point-to-point channels and interact with the external environment through four major I/O connections (pads). The data input stream is processed by the demux core (`dem`) that sends an audio stream to the audio decoder and a video stream to the video decoder. The video decoder accesses the external memory through a memory controller. The memory is used both as an intermediate storage and to send the decoded stream to the display controller and HDTV encoder. Finally, a master CPU controls the operation of all the blocks and handles the interaction with the environment. Additional non-functional constraints are often part of the specification: e.g, the `dem` core must occupy position $(0.2, 2.44)$ (in millimeter); the `cpu` communicates with the other cores, one at the time.

Figure 2 shows a *library of on-chip communication components* that contains a set of communication templates including

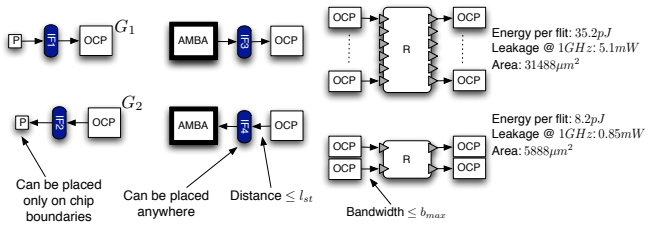


Fig. 2. A library of predefined communication components. Each component is characterized by performance and cost metrics.

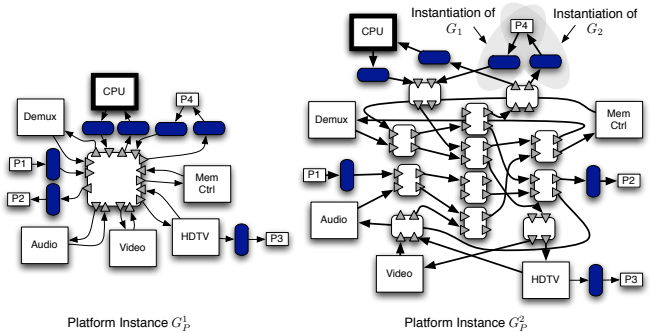


Fig. 3. Two NoC instances obtained by instantiation and composition of communication components.

interfaces IF1 and IF2 to connect pads with OCP cores, and interface IF3 and IF4 to connect AMBA cores with OCP cores. The library also contains various OCP routers that differ by the number of input and output ports. All these components are characterized by performance metrics, cost functions, and composition rules. Examples of these characterizations are: a link in a given metal layer can sustain up to a certain bandwidth $b_{max}$ and span a distance no greater than $l_{st}$, a parameterized router (taken from a pre-designed library) may not have more than a maximum number of I/O ports, and an IP core may feature only a specific protocol interface.

A communication structure that serves as the communication backbone for an SoC is constructed by *instantiating* communication templates (i.e. components from the library) and *composing* them. For instance, PAD4 is connected to the memory controller by instantiating the two communication templates $G_1$ and $G_2$. Figure 3 shows two alternative NoC implementations of the same specification. Network $G_P^1$ is obtained by instantiating the necessary interfaces plus a single $8 \times 8$ router while $G_P^2$ is obtained by instantiating only $2 \times 2$ routers. The performance and cost of the communication structure depends on the performance metrics and the cost functions of each component.

Our approach is detailed in the rest of the paper as follows: In Section II, we introduce formally the *SoC design specification* (i.e. the function), the target technology process with the library of communication components (combined, these represent a *platform* that captures all the admissible implementations), and the final communication implementation. In Section III, we show how to use this formal framework to formulate a general optimization problem for a class of libraries. In Section V-C, we provide a heuristic algorithm to solve this complex integer optimization problem. The algorithm is independent from the specific input constraints and the target platform. We do report a customization of the algorithm that takes into account bandwidth and latency constraints, expressed as hop count, to synthesize a minimal power network. The general algorithmic framework can be customized in several other different ways by changing cost function and constraints. Finally, in Section VI, we briefly describe the open-domain software infrastructure COSI-OCC (Communication Synthesis Infrastructure for On-Chip Communication) together with the results we obtain by applying it to a number of test cases that address the most challenging class of communication structure design: Network-on-Chip design. The open infrastructure can be used as a basis for developing new design flows by integrating additional synthesis and analysis tools [1]. The first release of COSI-OCC includes the following features:

- a set of synthesis algorithms that target different optimization criteria,
- supporting tools to simulate, validate, and visualize the synthesized network,
- a collection of communication libraries and SoC benchmarks, and
- extensive documentation.

## II. Setting the Stage: Definitions and Preliminaries

### A. Basic Definitions

The basic element of our formal framework is the *communication structure*. A communication structure is a set of interconnected components with associated *quantities* like, for instance, latency, bandwidth and position. A quantity $q$ takes on values from a domain $D_q$ that is partially ordered by a relation $\preceq_q$. We assume that $\perp$, which denotes no values, always belongs to the domain of a quantity $D_q$. Also, $\perp \preceq_q \nu$ for all $\nu \in D_q$.

Given a set of quantities $Q = \{q_1, \ldots, q_k\}$, the domain of $Q$ is the cross product $D_{q_1} \times \ldots \times D_{q_k}$. The domain of $Q$ is partially ordered by a relation $\preceq_Q$ point-wise induced by the relations $\preceq_{q_i}$. We use the same symbol $\perp$ to denote any tuple of $\perp$ values.

**Definition 1.** *A communication structure is a tuple $N(\mathcal{C}, Q, L)$ where $\mathcal{C} = \{c_1, \ldots, c_n\}$ is a set of components, $Q = \{q_1, \ldots, q_k\}$ is a set of quantities and $L \subseteq \{l | l : \mathcal{C} \rightarrow D_Q\}$ is a set of communication configurations. The set of components $\mathcal{C}$ is partitioned into two sets: the set of nodes $V \subseteq U_V$ and the set of links $E \subseteq V \times V$.*

The set $L$ of communication configurations captures the different ways in which quantities can be associated to components. The set $U_V$ is called the *node universe*. Similarly, the *component universe* is $U_{\mathcal{C}} = U_V \cup U_V^2$, and the *configuration universe* is $U_Q = \{l | l : 2^{U_C} \rightarrow D_Q\}$. Let $\mathcal{G}_Q$ denote the set of all communication structures with quantities $Q$.

[1]This approach is similar to the one our group followed in developing MIS that has been used for years as a platform to invent and test new logic synthesis algorithms.

**Example 1. (Communication structure):** Consider the set of quantities $Q = \{x, y\}$ representing the horizontal and vertical coordinates of a component. The domain $D_Q$ is the set of points where nodes can be placed. This domain can be described, for instance, by a discrete set of points or by union of rectangles. Elements of $D_Q$ are not comparable, therefore the order $\preceq_Q$ is a flat one, with $\perp$ being the minimum element. Given a communication structure $N(\mathcal{C}, Q, L)$, the set of configurations $L$ captures all the possible placements of the nodes in $V$. Since we do not assign any position to the links, for all $l \in L$ and for all links $e \in E$, $l(e) = \perp$. The additional constraint that no two nodes occupy the same position requires that for all $l \in L$, and for all pair of nodes $u, v \in V$, $l(u) \neq l(v)$.

We introduce two scoping operators on configurations. Given a communication structure $N(\mathcal{C}, Q, L)$, the restriction of a configuration $l \in L$ to a subset of components $\mathcal{C}' \subseteq \mathcal{C}$ is denoted by $l|_{\mathcal{C}'} = \{f : \mathcal{C}' \to D_Q | \forall c \in \mathcal{C}',\ f(c) = l(c)\}$. In particular, $l|_V$ and $l|_E$ are the restrictions of a configuration $l$ to the set of nodes and links, respectively. The restriction of a configuration to a subset of quantities $Q' \subseteq Q$ is denoted by $l[Q']$, and corresponds to projecting away the quantities not in $Q'$. We naturally extend these operators to sets of configurations. For instance, in Example 1, $L[x]|_V$ denotes the possible assignments of horizontal positions to nodes.

We use communication structures to capture three important and related concepts in our framework: the specification of an on-chip communication synthesis problem, the collection of alternatives to implement the communication, and the final communication implementation. Specifically:

- a *communication specification* as communication structure $N_C \in \mathcal{G}_{Q_C}$ where $Q_C = \{x, y, a, \tau, b, h\}$; nodes represent IP cores (that can be sources and/or destination of a communication) and have an associated position $(x, y)$ in the Euclidean plane, an area $a$, and a type $\tau$ denoting the supported interface protocol; links represent distinct inter-core communications. Each links is associated with two quantities: a minimum average bandwidth $b$ and a maximum latency $h$ (Section II-B);
- a *communication platform instance* as a communication structure $N_P \in \mathcal{G}_{Q_P}$ where $Q_P = \{x, y, \tau, in, out, \gamma\}$; $N_P$ contains either source/destination nodes or intermediate nodes, which correspond to components such as repeaters, routers and interfaces. Each node has an associate position $(x, y)$, a type $\tau$, two multisets $in$ and $out$ of input and output port interfaces, respectively. Each link is associated with a capacity $\gamma$, i.e. the maximum bandwidth that it can sustain (Section II-D);
- a *communication implementation* as a communication structure $N_I \in \mathcal{G}_{Q_I}$ where $Q_I = \{x, y, \tau, in, out, \rho, b, \gamma, h\}$; $N_I$ is obtained matching a communication specification onto a communication platform instance and adding to each intermediate node a non-empty transfer table $\rho$ (Section II-E). In addition to capacity and bandwidth, each link is characterized by a single output and input interface of the origin and destination node, respectively.

These three structures are described at different abstraction levels. Since the implementation $N_I$ is the result of matching the specification constraints, modeled by $N_C$, with the communication architecture capabilities, modeled by $N_P$, the set of quantities $Q_I$ contains more information than the sets $Q_C$ and $Q_P$. In Section II-E, we establish precise relations among the three structures to define when an implementation refines a platform instance and support a specification.

We use the following notation. For a given subscript $\sigma$, let $N_\sigma \in \mathcal{G}_Q$ be a communication structure. Then, we use $\mathcal{C}_\sigma$,$V_\sigma$, $E_\sigma$ and $L_\sigma$ to denote the sets of components, nodes, edges, and configurations of $E_\sigma$ respectively. We also implicitly assume that the set $Q$ contains a quantity $\lambda$ with domain $\Lambda$ representing a label attached to each component. For a component $c$, we denote its label with $\lambda(c)$.

### B. Communication Specification

We express an on-chip communication synthesis problem as a communication structure $N_C \in G_{Q_C}$. Each configuration $l \in L_C$ represents a possible combination of the positions and interfaces of the cores, and bandwidth and latency requirements for the communication among them (e.g., to capture different communication scenarios or different chip floor-planning).

**Example 2. (Communication specification):** In the set-top box example of Figure 1, the position of the $dem$ core is fixed at coordinates $(0.2, 1.44)$. Hence, each configuration $l \in L_C^{stb}$ must be such that $l(dem) = (0.2, 1.44, 0.55, OCP, \perp, \perp)$. Since there are no other floor-planning constraints, the position of the other IP cores can be determined during the synthesis process. Moreover, since the constraints between the CPU and the IP cores are mutually exclusive, for all $l \in L_C^{stb}[b]$, only one among $l(CPU, dem)$, $l(CPU, aud)$, $l(CPU, vid)$, $l(CPU, mem)$ can be different from zero.

Since the performance and cost of the network depend on the position of the cores, the first step in our design flow is to restrict the possible configurations of a specification by fixing the position of the ports of each core. In COSI-OCC we rely on the capabilities of the PARQUET floor-planner [15] to obtain these positions. More generally, given two communication structures $N_1$ and $N_2$ belonging to $\mathcal{G}_Q$, we say that $N_1 \subseteq N_2$ if $\mathcal{C}_1 = \mathcal{C}_2$ and $L_1 \subseteq L_2$.

### C. Communication Structures Instantiation and Composition

To allow the incremental design of complex on-chip communication, we introduce two operations: *renaming* and *parallel composition*. Renaming is used to change the identifiers of nodes so that two nodes in different sub-nets can be renamed to the same node, thus indicating an implicit connection between the two sub-nets at these nodes or indicating that the same IP is used to implement both. A *renaming function* $r : U_V \to U_V$ is a bijection on the vertex universe. With $R$ we denote the set of all renaming functions. Given a communication structure $N$ and a renaming function $r$, $r(N)$ denotes a new communication structure where the components have been renamed according to $r$.

The *composition* of two communication structures, denoted by $\|$, results in a new communication structure. The definition of the operator $\|$ embodies two rules. A rule is needed to establish how the configurations of the components being merged contribute to the formation of the ones of the combined entity. A second rule restricts the legal compositions by forcing the composed structure to satisfy certain properties. In general,

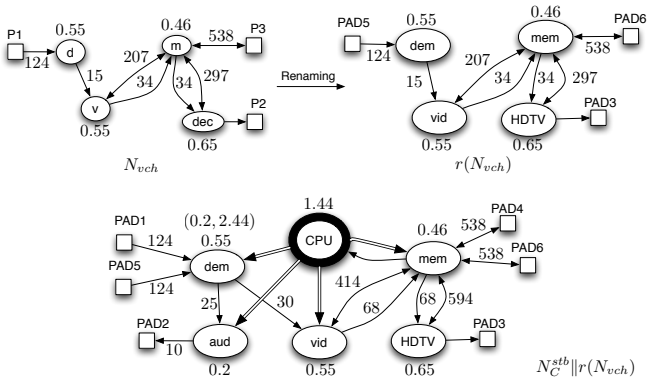


Fig. 4. Example of parallel composition of networks: the set-top box is expanded by adding a video channel and an extra off-chip memory bank.

the second type of rule is a relation between the components and the configurations of a communication structure.

**Definition 2.** *Given $N_1$ and $N_2$ belonging to $\mathcal{G}_Q$, their composition is $N_1 \|_Q^{\mathcal{R}} N_2 = N \in \mathcal{G}_Q$, where $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$, $L \subseteq L_1 \oplus_Q L_2$, $\oplus_Q$ is an associative and commutative operator on configurations, and such that $\{\mathcal{C}\} \times L \in \mathcal{R} \subseteq 2^{U_\mathcal{C}} \times U_Q$, i.e. composition rules $\mathcal{R}$ are satisfied.*

**Example 3. (Composition of communication specifications):** The operator $\oplus_Q$ depends on the quantities. A simple case is represented by the set $Q_C = \{x, y, a, \tau, b, h\}$. Assume that one extra video channel is added to our set-top box chip by *reusing* the already instantiated IP cores. In Figure 4, $N_{vch}$ is a communication structure capturing the communication requirements of a set-top-box video channel. To reuse the same IP cores, we rename the nodes according to a renaming function $r$ such that $r(d) = dem$, $r(m) = mem$, $r(v) = vid$ and $r(dec) = HDTV$. Since the new video channel must be displayed on the same device, $r(P2) = PAD3$ forces the same output pad to be reused. For the demodulator input, though, we need an additional pad. We also add a new pad to connect a second memory bank to the memory controller. Figure 4 shows the result of the composition $N_C^{stb} \|_{Q_C}^{\mathcal{R}_C} r(N_{vch})$. Intuitively, we have added the bandwidths of common requirements and we have restricted the position of the `dem` core. More precisely, given two communication structures $N_1,\ N_2 \in \mathcal{G}_{Q_C}$, the set $L = L_1 \oplus_{Q_C} L_2$ is such that:

- there is no "interference" between components not shared by $N_1$ and $N_2$, i.e $L|_{\mathcal{C}_1 \setminus \mathcal{C}_2} = L_1|_{\mathcal{C}_1 \setminus \mathcal{C}_2}$, and $L|_{\mathcal{C}_2 \setminus \mathcal{C}_1} = L_2|_{\mathcal{C}_2 \setminus \mathcal{C}_1}$;
- common components must be "compatible", i.e. they must agree on the positions and interfaces, i.e. $L[x, y, a, \tau]|_{V_1 \cap V_2} = L_1[x, y, a, \tau]|_{V_1 \cap V_2} \cap L_2[x, y, a, \tau]|_{V_1 \cap V_2}$ (notice that it is sufficient to have some compatible configurations for the composition to be non-empty);
- a configuration $l$ belongs to $L$ if only if there exist two configurations $l_1 \in L_1$ and $l_2 \in L_2$ s.t. for all $c \in E_1 \cap E_2$, $l[b](c) = l_1[b](c) + l_2[b](c)$ and $l[h](c) = \min\{l_1[h](c), l_2[h](c)\}$.

In this example we didn't use any additional composition rule ($\mathcal{R}_C$ contains all possible combinations of components and configurations). Instead, if a designer is given an area budget $\nu_a$ for the IP cores on a chip then the composition rule can be stated as follows:

$$\mathcal{R}_C = \left\{ (\mathcal{C}, l) \in 2^{U_\mathcal{C}} \times U_{Q_C} \;\middle|\; \sum_{c \in V} l[a](c) \leq \nu_a \right\}$$

We give examples of other rules in Section II-D.

### D. Libraries and Platforms

A platform is the set of all *valid* compositions that can be obtained by assembling the components from a given communication library. These components either have a corresponding implementation that is ready to be used or can be synthesized by tools operating at a lower level of abstraction.

A *communication library* $\mathcal{L}$ is a collection of communication structures, i.e. $\mathcal{L} \subset \mathcal{G}_Q$. The elements of a communication library, which are also referred to as components, are templates that can be instantiated and composed to obtain more complex communication structures. The set of quantities that characterizes our platform is $Q_P = \{x, y, \tau, in, out, \gamma\}$. Differently from $Q_C$, $Q_P$ represents the *capabilities* of a components. For instance, quantities $x$ and $y$ in $Q_C$ denote the coordinates where a component *must* be located, whereas the same variables in $Q_P$ denote the coordinates where a component *can* be located. Similarly, $\gamma$ is the maximum bandwidth that a link can support, i.e. the link *capacity*.

The definition of composition $\|_P^{\mathcal{R}_P}$ captures the set of valid communication architectures (i.e. communication platform instances) that can be obtained out of the communication library. The definition of the rules is more involved than the case described in Example 3 and depends on the design space that a designer is interested in. The following example shows the flexibility that our framework provides in defining the set of communication structures that can be obtained by composition of library elements.

**Example 4. Composition rules:** When modeling communication architectures at Layer 3 of the OSI model [16], a communication library contains nodes and links. Figure 5 shows a communication library $\mathcal{L}$ and two possible platform instances $N_P^1$ and $N_P^2$. The library contains the following set of components: a bus node and a bidirectional bus-segment connecting two bus nodes; a mesh node and two mesh links for East-West connection and North-South connection, respectively. The library also contains a set of interface communication structures to connect IP cores to bus nodes and mesh nodes. Each node has an associated multi-set of input interfaces $in$ and output interfaces $out$ (depicted as filled and non-filled shapes attached to nodes in Figure 5). A link connects an output interface of a node to an input interface of another node. Mesh links have an associated maximum capacity $\gamma_{max}^M$ while bus-segments (including the link between an IP core and a bus node) have an associated interval of capacities $[0, \gamma_{max}^B]$ corresponding to different configurations. We introduce two more quantities $i_x$ and $i_y$ that are the row and column index of a node in a mesh. The compositions rules can be stated as follows:

1) The number of bus nodes can be at most the number of bus segments minus 1. This condition ensures that the topology of a bus is a collection of trees. Also, since a bus node has only two bidirectional ports to connect to other bus nodes,each bus is a chain of IP cores (as shown by the platform instance $N_P^1$).
2) An East-West mesh link can connect two mesh nodes $(u, v)$ only if $l[i_x, i_y](u) = (i, j)$ and $l[i_x, i_y](v) = (i, j + 1)$; a North-South mesh link can connect two mesh nodes $(u, v)$ only if $l[i_x, i_y](u) = (i, j)$ and $l[i_x, i_y](v) = (i + 1, j)$ ( as shown by the platform instance $N_P^2$).
3) A bus is a connected chain of bus nodes. Each bus configuration $l$ must be such that the sum of the capacities of the links connecting the cores to the bus is less then $\gamma_{max}^B$. This rule restricts the possible configuration of a bus and models the fact that it is entirely shared among all IP cores connected to it.

These three rules define $\mathcal{R}_P$ for this specific platform.

**Definition 3.** *The* communication platform *generated by the*

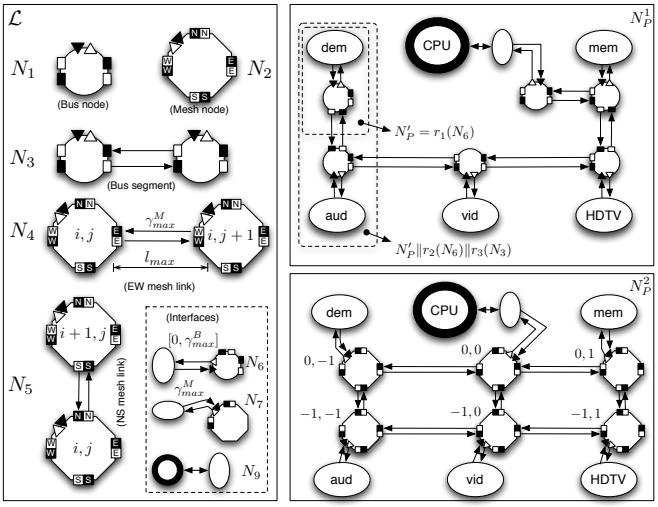


Fig. 5. Example of a library $\mathcal{L}$ and two alternative implementations for the set-top box based on composing elements instanced from $\mathcal{L}$.

*communication library $\mathcal{L}$ under composition $\|^{\mathcal{R}}_{Q}$ is*

$$\langle\mathcal{L}\rangle = \mathcal{L} \cup \quad \{ \quad N = N' \|^{\mathcal{R}}_{Q} N'_L : \\ N'_L \subseteq r(N_L), r \in R', \ N_L \in \mathcal{L}, \ N' \in \langle\mathcal{L}\rangle\}$$

*where $R' \subseteq R$ is a set of valid renaming functions. An element $N \in \langle\mathcal{L}\rangle$ is called a* communication platform instance.

### *E. Defining the Communication Implementation*

The implementation of a communication specification is a communication structure directly derived from a platform instance by adding the information regarding the routing of packets, captured by quantity $\rho$, and the latency, captured by $h$. Quantity $\rho$ is the transfer table associated to nodes. The domain of $\rho$ is $D_\rho = 2^{\Lambda\times\Lambda\times D_{out}}$. A transfer table is a set of triples $(\lambda_s, \lambda_d, o)$ where $\lambda_s$ and $\lambda_d$ are the labels associated with the source and destination of the packets, respectively, and $o$ is an output interface of a node. Each triplet specifies the proper output interface $o$ for each packet that arrives at the node from a given source in its transit to a given destination For routers, the transfer table is also called routing table.

Latency is associated to each link in a communication structure and in general is a quantity that is *derived* from the others. However, if it is measured in number of hops, then it is an independent quantity and each link has a latency equal to one. Another example of derived quantity is the bit error rate over wireless communication links that depends on the interference from other nodes in a communication structure. These quantities depend on the abstraction of the specific protocol that is used at the network level and at the lower level of abstraction (e.g., Layer 2 of the OSI protocol stack). For instance, packets travelling on a BUS incur in different latencies if the protocol is AMBA rather then OCP.

Let $q$ denote a derived quantity. Two cases can arise. If the configurations contain enough information to determine the value of $q$, then the quantity is *directly derived* from a function $m_q : U_Q[Q \setminus q] \times U_{\mathcal{C}} \rightarrow D_q$, and we call $m_q$ a *direct model*

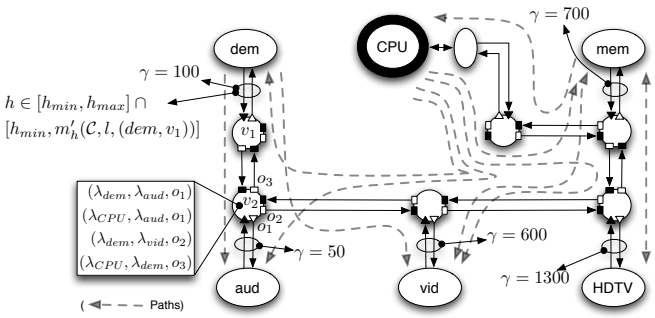


Fig. 6. Example of communication implementation for the set-top box.

for $q$. For instance, the power dissipated on a link is directly derived from its communication bandwidth. If the computation of the value of $q$ depends not only on the configuration but also on the components, then the quantity is *indirectly derived* from a function $m'_q : 2^{U_{\mathcal{C}}} \times U_Q[Q \setminus q] \times U_{\mathcal{C}} \rightarrow D_q$, and we call $m'_q$ an *indirect model* for $q$. When using an indirect model, the values of the derived quantities determined by operator $\oplus_Q$ are only bounded while the indirect model is part of the composition rules. For instance, the total capacity of the bus of Example 4 is captured as a composition rule.

In the case of latency, the same reasoning applies. Since the input-output latency of a router depends on the transfer tables, we can adopt a direct model for it. In the case of a bus, we can use bounds on the minimum latency and add an indirect model of latency in the composition rules to restrict the bus configurations only to the ones satisfying the model.

$\mathcal{R}_I$ contains the following important rule: for each configuration $l$ of a communication structure $N(\mathcal{C}, Q_I, L)$, $l[b] \leq l[\gamma]$, i.e. the bandwidth on each link must be less than or equal to the capacity of the link. An additional rule that can be added is *deadlock freedom* that requires the channel dependency graph [17] of $N$ to be acyclic.

**Example 5. Routing and latency:** Figure 6 shows a bus-based implementation of the set-top box example of Figure 1. The light-gray arrows represent paths in the communication structures. The paths are implicitly defined by the transfer tables of each bus-node. For instance, the transfer table of node $v_2$ contains an element $(\lambda_{CPU}, \lambda_{dem}, o_3)$ meaning that a packet from the `CPU` core to the `dem` core must be sent to output interface $o_3$. The transfer table of node $v_1$ contains an elements $(\lambda_{CPU}, \lambda_{dem}, o_1)$.

Assuming a 128 bit-wide bus and $200Mhz$ clock frequency, the maximum theoretical throughput is $1.6GB/s$. Hence, we can assign capacities to the links connecting the cores to the bus nodes as shown in figure. Given the capacity assignment, the communication implementation can support a larger set of specifications than the one in Figure 1. For instance, the throughput of the `dem` core can be increased to a maximum of $100MB/s$.

The latency to access the bus for each IP core depends on the actual set of components and the bus configuration. The latency of an end-to-end communication is the sum of the latencies of all components in the path. In this model we lump the latency on the access link to the bus and assign a latency equal to zero to each bus segment.

Given a communication platform instance $N_P$, there are many possible communication implementations $N_I \in \mathcal{G}_{Q_I}$ that can be obtained by associating transfer tables to the nodes in $\mathcal{C}_P$, i.e. by selecting the configurations $L_I$. The

communication implementation may or may not satisfy a given communication specification $N_C$. In fact, a configuration $l \in L_I$ defines one path for each end-to-end constraint. The cost of the implementation in terms of area occupation and power dissipation depends on the particular choice of the platform instance and its configuration. Hence, an algorithm is needed to find one path for each end-to-end communication constraint as well as the corresponding transfer tables so that the requirements are satisfied.

Formally, a path of length $n$ is a sequence of $n$ links $\pi = (e_1, \ldots, e_n)$ such that $e_i = (v_{i-1}, v_i)$. A *real path* from a source node $s$ to a destination node $d$ according to a configuration $l$ is such that $v_0 = s$, $v_n = d$ and $\forall e_i \in \pi,\ \exists(\lambda_s, \lambda_d, o) \in l[\rho](v_{i-1})$ and $l[out](e_i) = o$. Given an implementation $N_I$, $\Phi(N_I)$ denotes an implementation derived from $N_I$ by removing links and nodes not belonging to any real path for any configuration. We introduce an order among communication structures to establish a ranking that depends on the quantities associated to the components.

**Definition 4.** *Given two communication structures $N_1, N_2 \in \mathcal{G}_Q$, $N_1 \leq_Q N_2$ if and only if $\mathcal{C}_1 \subseteq \mathcal{C}_2$, and for all $l_1 \in L_1$ there exists $l_2 \in L_2$ such that for all components $c \in \mathcal{C}_1$, $l_1 \preceq_Q l_2|_{\mathcal{C}_1}$.*

The partial order $\preceq_Q$ is induced by the partial order of each quantity in $Q$. For instance, $\preceq_b$ is the natural order $\leq$ defined on real numbers. The communication specification characterizing the set of specifications that a communication implementation $N_I$ can correctly implement is given by the *path abstraction* $\Pi : \mathcal{G}_{Q_I} \rightarrow \mathcal{G}_{Q_C}$, which is defined by the following construction:

- the nodes of $N_C$ are the IP cores also present in $N_I$;
- there exists a link $(s, d) \in \mathcal{C}_C$ if and only if there exists a configuration $l_I \in L_I$ such that is a real path from $s$ to $d$ in $N_I$ according to $l_I$;
- a configuration $l_C$ belongs to $L_C$ if an only if there exists a configuration $l_I \in L_I$ such that the following conditions are satisfied:
  1) $l_C[x, y, \tau] = l_I[x, y, \tau]|_{\mathcal{C}_C}$
  2) for all links $e \in \mathcal{C}_I$
$$\sum_{(s,d)\in\mathcal{C}_C : e\in\pi(s,d)} l_C[b](s, d) = l_I[b](e)$$
  3)
$$\sum_{e\in\pi(s,d)} l_I[h](e) = l_C[h](s, d)$$

To verify that $N_I$ is a refinement of a communication platform instance, we introduce an abstraction relation $\Psi : \mathcal{G}_{Q_I} \rightarrow \mathcal{G}_{Q_P}$ that simply removes the transfer tables and the latency quantities, i.e. given an implementation $N_I$ it returns $\Psi(N_I) = N_P(\mathcal{C}_I, Q_P, L[Q_P])$. Then, given a specification $N_C$ and a platform $\langle\mathcal{L}\rangle$, the implementation $N_I$ must satisfy the following constraints: $N_C \leq_{Q_C} \Pi(N_I)$ and $\Psi(N_I) \in \langle\mathcal{L}\rangle$. When the implementation is constrained to have a specific topology such as a mesh or a torus, a more stringent constraint $\Psi(N_I) \leq_{Q_P} N_P$ must also be satisfied where $N_P$ captures also the topology.

## III. Formulation of the Optimization Problem

For a given specification $N_C$ and platform instance $N_P$, our objective is to find an implementation $N_I$ that minimizes a given cost function $F(N_I)$. The communication synthesis problem can be stated as follows:

$$\texttt{PR1}: \quad \min_{L_I} \quad F(N_I) \qquad \textit{subject to} \quad N_C \leq_{Q_C} \Pi(N_I),\ \Psi(N_I) \subseteq N_P$$

where $F : \mathcal{G}_{Q_I} \rightarrow R_+$ is a cost function such that $F(N_I) = F(\Phi(N_I))$, i.e. the cost of a component is accounted for only if it is actually used in the implementation. This approach has been adopted in the optimization of NoC with fixed topologies where, for instance, $N_P$ is a mesh [8].

Let $Alg$ be a hypothetical algorithm that solves problem PR1 exactly. Given a library $\mathcal{L}$, platform $\langle\mathcal{L}\rangle$ can be explored by using $Alg$ to solve problem PR1 for each $N_P \in \langle\mathcal{L}\rangle$. In [7], the optimization problem is solved for many instances corresponding to meshes, tori, butterflies and other regular topologies. In [18], the optimization technique explores the isomorphic-free set of all regular topologies.

**Lemma 1.** *Let $N_C$ be a specification, $N_{P,1}$ and $N_{P,2}$ two platform instances such that $N_{P,1} \leq N_{P,2}$. Let $N^*_{I,1}$ and $N^*_{I,2}$ be the implementations found by Alg for platform instances $N_{P,1}$ and $N_{P,2}$, respectively. Then $F(N^*_{I,2}) \leq F(N^*_{I,1})$.*

*By contradiction.* . Suppose that $F(N^*_{I,1}) \leq F(N^*_{I,2})$. Since $\Psi(N^*_{I,1}) \subseteq N_{P,1} \leq N_{P,2}$ we can construct an implementation $N_{I,3}$ such that $\Psi(N_{I,3}) \subseteq N_{P,2}$ and $N^*_{I,1} \leq N_{I,3}$ (by instantiating more components in $N^*_{I,1}$ or, if $N_{P,1}$ and $N_{P,2}$ have the same set of components, by increasing the capacities of some links). The set of configurations of $N_{I,3}$ is such that $L_{I,3}[Q \setminus \gamma]|_{\mathcal{C}^*_{I,1}} = L^*_{I,1}$, and $L_{I,3}[Q \setminus \gamma]|_{\mathcal{C}_{I,3}\setminus\mathcal{C}^*_{I,1}} = \perp$ otherwise. Hence, we have that $\Phi(N^*_{I,1}) = \Phi(N^*_{I,3})$ and consequently $N_C \leq \Pi(\Phi(N^*_{I,1})) = \Pi(\Phi(N_{I,3}))$ which means that $N_{I,3}$ satisfies the specification. By the hypothesis on the cost function we obtain:

$$F(N_{I,3}) = F(\Phi(N_{I,3})) = F(\Phi(N^*_{I,1})) = F(N^*_{I,1}) \leq F(N^*_{I,2})$$

which contradicts the optimality of $Alg$. □

According to Lemma 1, if we can find the greatest element $\overline{N_P}$ of $\langle\mathcal{L}\rangle$ with respect to the ordering relation $\leq$, then the solution of problem PR1 with $N_P = \overline{N_P}$ is the best communication structure among all possible platform instances. Unfortunately, such greatest element is not guaranteed to exist in any given platform. Instead of looking for the greatest element, we can look for an upper bound $N_P^{\langle\mathcal{L}\rangle}$ of $\langle\mathcal{L}\rangle$ and add the following constraint to the optimization problem:

$$\Psi(\Phi(N_I)) \in \langle\mathcal{L}\rangle$$

### A. Construction of the upper bound of the communication platform and formulation of the optimization problem

We focus our attention to the construction of an upper bound $N_P^{\langle\mathcal{L}\rangle}$ for any given library of nodes and links. Because we want any configuration $l_I[x, y, \tau]$ to be injective, the maximum number of nodes in any platform instance is limited

to $|D_{\{x,y,\tau\}}|$. Thus, let $U'_V \subset U_V$ be a set of nodes such that $|U'_V| = |D_{\{x,y,\tau\}}|$. The set of valid renaming functions is such that any renaming of the nodes of the library elements is in $U'_V$.

**Proposition 1.** *If $|D_{\{x,y,\tau\}}|$ is finite, there exists an upper bound $N_P^{\langle\mathcal{L}\rangle} \in \mathcal{G}_{Q_P}$ of the communication platform $\langle\mathcal{L}\rangle$ with respect to the ordering relation $\preceq_{Q_P}$.*

*By construction.* Let $N_P^{\langle\mathcal{L}\rangle}$ be the communication platform instance defined as follows. The set of components is $\mathcal{C}_P^{\langle\mathcal{L}\rangle} = U'_V \cup U'_V \times U'_V$. The set $L_P^{\langle\mathcal{L}\rangle}$ contains all configurations $l$ such that $l[x,y,\tau]|_{V_P^{\langle\mathcal{L}\rangle}}$ is injective; for all links $e(u,v) \in E_P^{\langle\mathcal{L}\rangle}$, $l(e) = q$ if and only if there exists a component $N_L$, a valid renaming function $r \in R'$ and a configuration $l_L \in L_L$ such that $l_L(r^{-1}(u)) = l(u)$, $l_L(r^{-1}(v)) = l(v)$ and $l_L(e_L) = q$, and $q = \perp$ otherwise. The upper bound has the following property, called *maximum capacity*. For all possible assignment of positions and types of the nodes, there is a configuration $l$ such that for each link $e \in \mathcal{C}_P^{\langle\mathcal{L}\rangle}$ that is an instance of a component $N_i$, $l[\gamma](e) \geq \max\{L_i[\gamma](r^{-1}(e))\}$. A platform instance $N_P$ can be written as $N_P = r_1(N_1)\|_{Q_P}^{\mathcal{R}_P} \cdots \|_{Q_P}^{\mathcal{R}_P} r_k(N_k)$, for $r_i \in R'$ and $N_i \in \mathcal{L}$. Because of the restriction on the validity of the renaming functions, $\mathcal{C}_P \subseteq \mathcal{C}_P^{\langle\mathcal{L}\rangle}$. Moreover, since rule $\mathcal{R}_P$ requires that all configurations $l[x,y,\tau] \in L_P[x,y,\tau]$ be injective, for each of them there exists a configuration $l' \in L_P^{\langle\mathcal{L}\rangle}$ such that $l[x,y,\tau] \preceq_{Q_P} l'[x,y,\tau]|_{\mathcal{C}_P}$. Given the maximum capacity property of the upper bound, for such assignment of position and type of nodes there is a configuration $l'' \in L_P^{\langle\mathcal{L}\rangle}$ such that for all $e \in E_P$, $l''[\gamma](e) \geq l[\gamma](e)$. Therefore $l \preceq_{Q_P} l''|_{\mathcal{C}_P}$. From the generality of $N_P$, follows that $N_P^{\langle\mathcal{L}\rangle}$ is an upper bound of $\langle\mathcal{L}\rangle$. □

**Corollary 1.** *$N_P^{\langle\mathcal{L}\rangle}$ as constructed in the proof of Proposition 1 is the least upper bound of $\langle\mathcal{L}\rangle$.*

Finding the *least* upper bound is important because the complexity of the exploration depends on the cardinality of the set of vertices, links and labeling sets. The least upper bound minimizes such cardinality and it is, therefore, a good candidate input to an efficient heuristics. From Lemma 1 and Proposition 1, to solve the communication synthesis problem, we need to solve the following optimization problem:

$$\begin{aligned} \texttt{PR2}: \quad & \min_{L_I} \quad F(N_I) \\ & \textit{subject to} \quad N_C \leq \Pi(\Phi(N_I)),\ \Psi(N_I) = N_P^{\langle\mathcal{L}\rangle}, \\ & \qquad\qquad \Psi(\Phi(N_I)) \in \langle\mathcal{L}\rangle \end{aligned}$$

In general the upper bound $N_P^{\mathcal{S}}$ does not satisfy the composition rules $\mathcal{R}_P$ (in fact, these rules are not taken into account by the constructive proof of the upper bound itself). The additional constraint $\Psi(\Phi(N_I)) \in \left\langle \hat{\mathcal{L}} \right\rangle$ makes sure that the implementation satisfies the composition rules, and could, indeed, be replaced by $\Psi(\Phi(N_I)) = N_P$ and $\{\mathcal{C}_P\} \times L_P \subseteq \mathcal{R}_P$. If $N_I^*$ is the optimal solution, $\Phi(N_I^*)$ is the optimal communication implementation.

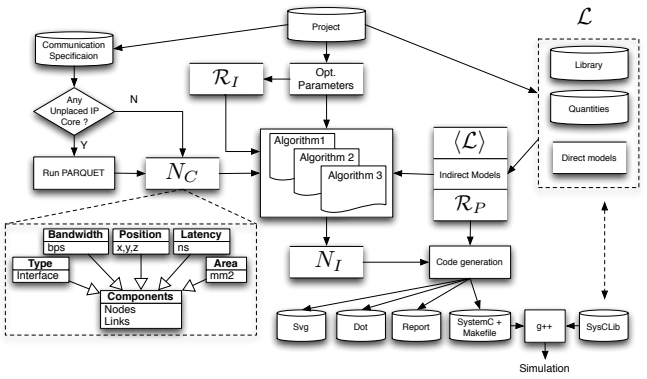


Fig. 7. COSI-OCC open software infrastructure.

## IV. The COSI-OCC Open Software Infrastructure

Figure 7 shows the software organization of COSI-OCC. The input to COSI-OCC is a project file that contains pointers to the communication specification and to the library. All files are in XML format. The communication specification contains a list of IP cores and inter-core communication constraints. If there are unplaced IP cores, PARQUET is used to floor-plan the chip [15]. The library file contains the description of each library element, the quantities attached to them, and the parameters needed to compute the value of directly derived quantities. The library is used to construct the platform data structure that also contains the composition rules including models for indirectly derived quantities. The project file includes also the optimization parameters such as the relative weight of power and area cost. The communication specification and the platform are passed to the synthesis algorithm that derives the network implementation $N_I$.

COSI-OCC includes a set of code generators to produce an SVG graphical representation and a DOT logical representation of $N_I$. A SYSTEMC netlist can be generated from $N_I$ by assembling the corresponding SYSTEMC-view of each element instanced from the library that is contained in `SysCLib`, also part of the COSI-OCC distribution. The generation of the SYSTEMC netlist is a further refinement of $N_I$ that requires the binding of each port of the nodes to links, the generation of the routing tables, and the computation of the weights for the *weighted fair queuing* algorithm, which is used by the routers to schedule flits. The COSI-OCC distribution includes a set of algorithms to solve some variants of the communication synthesis problem. For instance, we provide an algorithm that generates deadlock-free networks, e.g. for cases where support for virtual channels is not available in the routers. Our approach to solve this problem is different from the one proposed in [11] where a graph with all possible connections is constructed explicitly (i.e the Switch-Cost Graph) and the set of prohibited turns is precomputed before deciding the routing of packets. In COSI-OCC, we build the solution incrementally by instantiating links from the library. The optimization algorithm operates directly on the channel-dependency graph of the communication structure and at run-time checks that such graph is kept acyclic (i.e. it checks that the corresponding composition rule is satisfied). More information on COSI-OCC is available at the COSI project web site [19].

## V. Application to Network-on-Chip Synthesis

In this section we apply our methodology to the synthesis of NoCs. We develop models for nodes and links and we use them to define the set of configurations of the library elements. The models include the cost of each element in terms of area and power consumption. We define the composition rules and develop an efficient heuristic algorithm to solve problem PR2.

### A. Modeling Nodes and Links

Figure 8(a) shows the internal architecture of an input-queued router. Data flits arriving at the inputs are buffered into queues and then sent to the right output by a cross-bar switch according to the information stored in a routing table. [2] Depending on the number of virtual channels supported by the router, there can be one or more queues for each input, called lanes. A router with $i$ inputs and $j$ outputs is characterized by an energy-per-flit metric $E(i,j)$ and an area metric $A(i,j)$. The table in Figure 8(b) reports $E(i,j)$ values across different router configurations and technology processes. Given a target technology process, the area and energy dissipation of a router depend on five parameters: number of inputs, flit-width, number of lanes, queue length inputs, and number of outputs. We obtained these values not through an analytical model, but by running a series of simulations with ORION [20].

Network interfaces are directly connected to cores. Their characterization in terms of power and area is the same as for the routers. Differently from routers, interfaces need to provide extra services such as protocol conversion, flit-width adjustment, and packetization. Hence, their performance can be very different from the one of a router especially in terms of throughput and latency. We are aware of this difference and we plan to incorporate more detailed models for network interfaces in future release of COSI-OCC.

A link is a bundle of wires that connects the output port of a node with the input port of another node. Figure 8(c) shows the first-order RC model of a buffered wire where: $R_d$ is the transistor driving resistance, $w$ is the width of the buffer NMOS transistor normalized by the minimum technology width, $\beta$ is the PMOS-to-NMOS sizing ratio, $C_d$ and $C_g$ are the diffusion and gate capacitance per unit width, $R_w$ and $C_w$ are the wire resistance and capacitance per unit length, and $l_{sg}$ is the length of the buffered segment. The delay $d$ of such segment is :

$$d = 0.7\left[\frac{R_d}{w}\cdot C_1 + (R_w + C_w)\frac{l_{sg}^2}{2} + l_{sg}\cdot R_w \cdot w(\beta+1)C_g\right]$$

where $C_1 = (w(\beta+1)(C_d+C_g) + l_{sg}\cdot C_w)$. To make the delay linear in the wire length, designers can insert an optimal number of buffer with optimal-size $w^\star$ spaced by a distance $l_{sg}^\star$ called *critical length* [21]–[23]. The delay $d^\star$ of a critical length is called *critical delay*. Assuming a synchronous design implementation with target clock frequency $f$, the maximum distance that a signal can travel on an optimally repeated wire within one clock period $T = 1/f$ is $l_{st} = \left\lfloor \frac{T}{d^\star}\cdot l_{sg}^\star \right\rfloor$ and it is called *critical sequential length*.

[2] We assume that the bit-width of each port is equal to the flit-width. The routing table and scheduler are not shown in Figure 8.

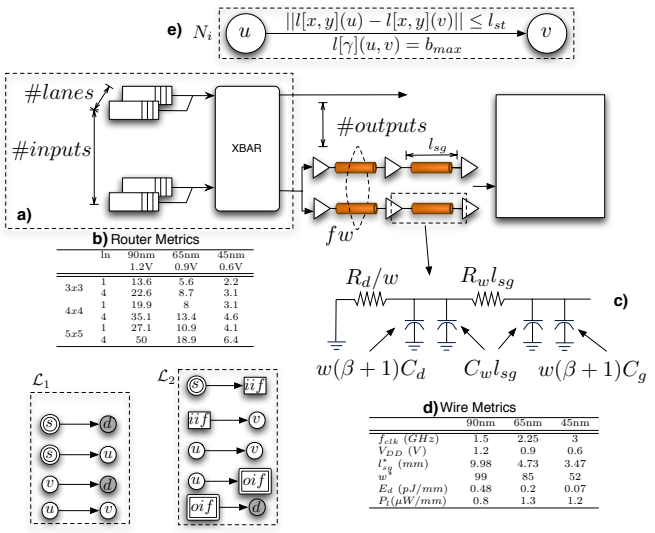


Fig. 8. Modeling the NoC components.

The power dissipated by an optimally-buffered line of length $l$, where $l_{sg} < l < l_{st}$, running at frequency $\frac{1}{T}$ with an activity factor $\alpha$ (the fraction of buffers that are switched during an average clock cycle) is:

$$P = \frac{l}{l_{sg}^\star}\cdot\left[\frac{\alpha}{T}V_{dd}^2\cdot(C_1^\star + C_w) + \frac{V_{dd}}{2}\cdot\left(w^\star(I_n^{off} + 2I_p^{off})w_n^{min}\right)\right]$$

where $V_{dd}$ is the supply voltage and $I_n^{off}$ ($I_p^{off}$) is the leakage currents per unit NMOS(PMOS), and $w_n^{min}$ is the width of the NMOS transistor of a minimum-size inverter. The two terms of the sum are the switching power and the leakage power. [3]

The table in Figure 8(d) summarizes the metrics of interests for the purpose of NoC synthesis. In particular, each link is characterized by an energy dissipation per bit per unit length $E_d/l$ and an area per bit per unit length, which includes the wiring and buffer areas.

### B. The Communication Library and the Composition Rules

Figure 8(e) shows the basic NoC component $N_i$, i.e. a link. The set of configurations of a component contains all assignments of positions to the two nodes such that their distance is not greater than the maximum distance $l_{st}$. The capacity of a link is equal to $b_{max}$ and the latency is equal to one hop. The capacity $b_{max}$ is different from the clock frequency. In fact, in order to avoid router congestion, the capacity of a link should be set in such a way that the routers' injection rate is far from saturation. Otherwise, the actual communication latency would grow exponentially.

In Figure 8, $\mathcal{L}_1$ and $\mathcal{L}_2$ are two possible communication libraries. There are many types of nodes: $s$ is a source node (withou any input), $d$ is a destination node (without any output), $u$ and $v$ are routers, $iif$ is an input interface and $oif$ is an output interface. Since each component in $\mathcal{L}_1$ has the same interface, this library allows establishing

[3] We simplified the formula omitting the contribution of the short-circuit current which is negligible with respect to the other two contributions.

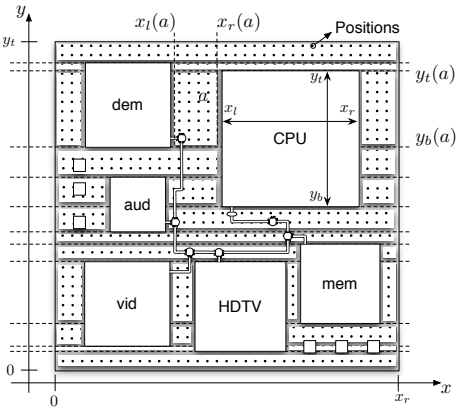


Fig. 9. Slicing method to find the available area for NoC implementation.

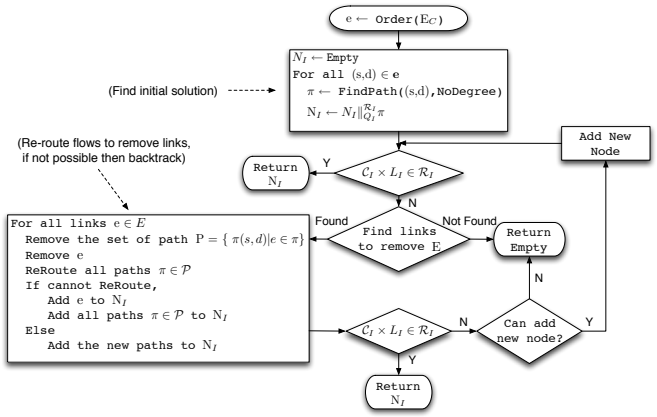


Fig. 10. High-level description of the heuristic algorithm.

direct connections between a source and a destination. Instead, library $\mathcal{L}_2$, where the source interface $iif$ is different from the destination interface $oif$, supports a design flow where there are dedicated sockets to connect the cores to the NoC.

Two important composition rules are implemented in COSI-OCC. At the platform level, rule $\mathcal{R}_P$ allows only communication structures where the number of input and output links of a node does not exceed the number of input and output ports, respectively. At the implementation level, rule $\mathcal{R}_I$ allows only deadlock-free communication structures by forcing the channel-dependency graph of an implementation to be acyclic. Moreover, the bandwidth on any channel cannot exceed its capacity.

### C. Solution to the Optimization Problem

As described in Section II, we begin our optimization process by assigning a fixed position to all cores with a floorplanner. To identify some of the degrees of freedom of the optimization variables, we have to define the area $A_C$ available to lay out the network. We assume here that placing the network over the cores is not allowed. Therefore, $A_C$ is the area of the the chip that is not occupied by the IP cores. Figure 9 shows the floor-plan of the set-top box example and a possible layout of the bus implementation of Figure 6. $A_C$ can be represented as the union of a finite set of rectangles $\mathcal{A}$ that can be automatically computed. Each rectangle $a \in \mathcal{A}$ is described by four real numbers $x_l(a)$, $x_r(a)$, $y_t(a)$ and $y_b(a)$, denoting its left, right, top and bottom boundaries, respectively. Next, to apply Proposition 1, we need to discretize the rectangles to make $|D_{x,y,\tau}|$ finite. Given a parameter called density $\delta$, we assume that the available positions are uniformly distributed in $a$. Hence the number of positions available is equal to $\lceil \delta \cdot (x_r(a) - x_l(a))(y_t(a) - y_b(a)) \rceil$. The set $D_{\{x,y\}}$ is the union of the positions in all rectangles in $\mathcal{A}$.

At this point, we may attempt at solving Problem PR2 with Integer Linear Programming (ILP). To illustrate this approach, consider library $\mathcal{L}_1$ of Figure 8. First, we have to linearize the cost function by assuming that the cost of a router is the sum of the cost of each input port plus the cost of each output port. Then, we define the energy per flit as $\min_{i,j}[E(i+1,j) - E(i,j)]$ for an input port and as $\min_{i,j}[E(i,j+1) - E(i,j)]$ for an output port (and, similarly, the leakage power and area occupation). This linearized cost function is a lower bound of the real cost of the network. Hence solving the ILP with this cost function will give us a solution that is optimistic. Using one binary variable for each installation site denoting whether a router is installed at that site, one binary variable for each link that can be installed between two sites, and one binary variable to denote that a constraint is routed through a link, the number of variables of the ILP problem becomes very large. It is equal to $|\hat{D_{xy}}|^2 \cdot |E_C| + |\hat{D_{xy}}|^2 + |\hat{D_{xy}}|$where the first term is the square of the number of installation sites times the number of constraints. For the simple example of Figure 1 with 70 installation sites, the number of binary variables is $93,170$. These many variables cause an ILP solver to run very slow. Moreover, some composition rules (e.g deadlock freedom) cannot be included in the ILP since they are highly non linear. Because of these difficulties, we devised a heuristic approach to solve problem PR2. In Section VI, we will compare the results obtained by the heuristic with a lower bound provided with a further optimistic approximation of the ILP formulation.

*1) Structure of the Algorithm:* Figure 10 shows the high-level structure of the heuristic algorithm. In the first step, we find an initial solution not taking into consideration node degree constraints. In the second, we use an iterative procedure that removes degree violations by deleting links and/or adding routers. The initial solution is found with the same technique that is used in algorithms for global routing: the end-to-end constraints in $E_C$ are first ordered by decreasing bandwidth. One path in $N_P^{\langle \mathcal{L} \rangle}$ is then found for each constraints one at a time (the actual implementation of procedure FindPath depends on the composition rules). In this phase, the degree constraint rule is not taken into account; however, if we are lucky, $N_I$ may still satisfy the degree rule, in which case the algorithm returns $N_I$ and stops. Otherwise, we activate an iterative procedure to remove the degree constraint violations.

This procedure implements a rip-up and reroute approach one link at a time. The links connected to the output of nodes with output degree violations and links connected to the input of nodes with input degree violations are the ones that are considered for rip-up and re-route. For each link, all

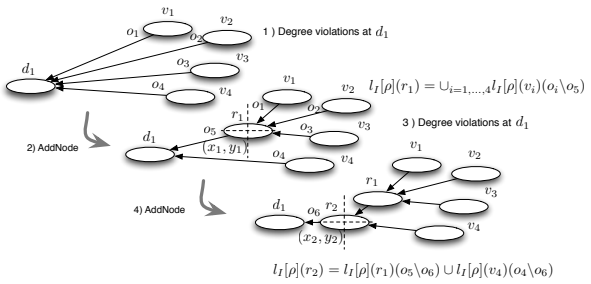


Fig. 11. Procedure for adding a new router to the NoC implementation. For an expression $exp$, we denote by $exp(x \backslash y)$ the same expression where variable $x$ has been replaced by $y$.

source-destination paths containing that link are re-routed by procedure `FindPath` that now takes into account the degree constraint rule. If a path cannot be removed, the algorithm back-tracks by reinserting the link and all the paths. Otherwise, the new paths are added to the communication implementation. If the re-routing procedure finds an implementation that satisfies the composition rules, the algorithm ends with success.

Otherwise we try adding a new node (router) to yield a feasible solution. The idea is that when a new node is added, multiple links entering/exiting a node can be merged/split into/from one link, thereby reducing the degree of the node (Figure 11). However, if no node can be added (e.g., because delay constraints would be violated) the algorithm ends with an empty implementation implying that no solution was found. Figure 11 illustrates how new nodes are added. First, among all nodes with input/output degree violations, the one with the highest number of input/output links is selected. All input/output links to/from the node are candidates for the merge operation. A subset of them is chosen with a criterion that depends on the optimization goal. The source and target nodes of the selected links are connected to the new router, which is instanced in a position such that the cost of the links is minimized. The transfer table of the router is set according to the new paths flowing through it. As it executes this local transformation the algorithm makes sure that link capacities and degree constraints are not violated. Note that the merge operation does not change the number of nodes with degree violations.

*2) The* `FindPath` *procedure:* This procedure is available in different forms to search for the "best" path between a source and a destination core depending on the particular composition rules that the user specified. If a delay model must be taken into account to check delay constraints (rule $\mathcal{R}_1$), the best path is discovered by a labeling algorithm (`SpLabeling`) that finds the minimum-cost constrained shortest path between two nodes; a modified version of the Dijkstra shortest path algorithm is used otherwise. If deadlock freedom (rule $\mathcal{R}_2$) is included in the set of rules $\mathcal{R}_I$, then `FindPath` runs on the channel-dependency graph of the communication implementation to make sure that this graph remains acyclic. The degree constraints of the nodes can be taken into account by adding rule $\mathcal{R}_3$).

`FindPath` explores the upper bound $N_P^{\langle\mathcal{L}\rangle}$ without building an explicit representation. In fact $N_P^{\langle\mathcal{L}\rangle}$ is explored locally

**Procedure** `Reach` $(N_I, v, \mathcal{R}, \mathcal{L})$

$N_R \leftarrow$ empty communication structure ;
**forall** $N_i \in \mathcal{L}$ *with* $C_i = \{v_i, u_i, (u_i, v_i)\}$ **do**
  **forall** $l_i \in L_i$ **do**
1  **if** $l_I(v) = l_i(u_i)$ **then**
2  let $N_i'(\mathcal{C}_i, Q_P, \{l_i\})$ ;
3  $N_i'' \leftarrow r(N_i)$, with $r(u_i) = v, r(v_i) = id(l_i(u_i))$ ;
4  **if** $\Psi(N_I)\|_Q N_i''$ *satisfies* $\mathcal{R}$ **then**
  $N_R \leftarrow N_R\|_Q N_i''$
**return** $N_R$ ;

at run-time by procedure `Reach`. This procedure takes as input parameters the current communication implementation $N_I$, a node $v \in \mathcal{C}_I$, the composition rules $\mathcal{R}$, and the platform library $\mathcal{L}$. `Reach` checks which links can be instantiated with the source node $v$ (Line 1). For each link, an instance is generated by renaming the nodes appropriately (Lines 2 and 3). Function $id$ associates a unique identifier to a node depending on its type and position. If the new link can be composed with the communication implementation without violating the composition rules (Line 4), then it is added to the reachable communication structure $N_R$ (note that $N_R \in \mathcal{G}_{Q_P}$. Therefore the set of rules $\mathcal{R}$ must contain $\mathcal{R}_P$).

**Procedure** `SpLabelling` $(s, d, N_I, \mathcal{L}, \mathcal{R}, \mathcal{R}_I)$

1 $D[s] \leftarrow \{(0,0)\}$, $D[v] \leftarrow \emptyset$, $\forall\, v \in U_V' \setminus \{s\}$ ;
2 $Q \leftarrow (s, (0,0))$ ;
**while** $Q \neq \emptyset$ **do**
3  $(v, D_v) \leftarrow$ `ExtractMin` $(Q)$ ;
4  $N_\pi \leftarrow$ path from $(s, (0,0))$ to $(v, D_v)$ ;
  $N_R \leftarrow$ `Reach` $(N_I\|_{Q_I} N_\pi, v, \mathcal{R}_P, \mathcal{L})$ ;
5  **forall** $(v, u) \in \mathcal{C}_R$ **do**
  $l \leftarrow$ `Configure` $(L_I, (v, u))$ ;
  $N' \leftarrow (\{(u, v)\}, Q_I, \{l\})$ ;
6  **if** $N_I\|_{Q_I} N'$ *satisfies* $\mathcal{R}_I$ **then**
7  $f \leftarrow$ Compute incremental cost $\Delta F$ ;
8  $D_u = (D_v.H + 1, D_v.C + f)$ ;
  **if** $\nexists D \in D[u]$ *s.t.* $D < D_u$ **then**
9  $D[u] \leftarrow D[u] \cup \{D_u\}$ ;
10  `Insert` $(Q, (u, D_u))$ ;
11  $\pi[(u, D_u)] \xleftarrow{N'} (v, D_v)$ ;
12 **if** $D[d] = \emptyset$ **then**
  **return** $\emptyset$ ;
**else**
  **return** `ToGraph` $(\pi)$ ;

Procedure `SpLabelling` is one particular implementation of `FindPath`. It solves the constrained shortest-path problem [24] using a labeling algorithm [25]. We use the number of hops as a model for latency. A distance label is a tuple $D = (H, C)$ associated to a node $v$ where $H$ is the number of hops of the path from the source $s$ to $v$ with minimum cost $C$. A distance label $D$ is dominated by $D'$, written $D < D'$ if $D.H \leq D'.H$, $D.C \leq D'.C$, and $D.H \neq D'.H \vee D.C \neq D.C'$. A set of distance labels $D[v]$ is associated to each node $v$. The queue $Q$ contains pairs $(v, D)$ where $v$ is a node and $D \in D[v]$ is a distance label of $v$.

Distance labels in the queue are ordered by number of hops and for the same number of hops, by cost. The procedure starts with an empty set of distance labels for all nodes but the source, which has the distance label $\{0, 0\}$. The pair $(s, \{0, 0\})$ is the only element in the queue (Line 2). The minimum distance label node is extracted from the queue (Line 3) and the set of possible links departing from the node (computed by procedure `Reach`) is processed (Lines 4 and 5). Each link is first configured by selecting one possible configuration, then composition rules are checked (Line 6). If this distance label is not dominated by any other already present at $u$, then it is added to the set of distance labels of $u$ (Line 9), the new pair is added to the queue (Line 10), and the predecessor tree is updated (Line 11). A path from $s$ to $d$ that satisfies the hop constraint exists if the set of distance labels at $d$ is not empty (Line 12). If this is the case, the path with minimum cost $C$ is selected and returned. During the construction of the initial solution, the composition rules $\mathcal{R}$ and $\mathcal{R}_I$ do not contain rule $\mathcal{R}_3$, which is added during the re-routing procedure instead.

## VI. Experimental Results

Table I lists the SoCs that we used in our experiments. We selected the test cases based on several criteria:

- the number of IP cores $|V_C|$, ranging from 12 to 42, and the size of the chip, as large as $48mm^2$;
- the total bandwidth requirement, defined as the sum of the bandwidth requirements over all end-to-end constraints $E_C$, and ranging from 9 to 99 $Gbps$;
- the maximum input degree of a destination core and the maximum output degree of a source core, ranging from 2 to 25 depending on the SoC application.

The goal of our experiments is to study the impact of these application features on the synthesized NoC. Specifically we are interested in the following metrics: the power and area breakdown, the maximum and average input and output degree of the nodes , the maximum and average number of hops among all source-destination paths, and the maximum and average latency. The latency measurements are obtained by simulating the SystemC implementation of the NoC generated by COSI-OCC. The maximum latency is the largest end-to-end delay experienced by any packet over the entire simulation, i.e. the time that elapses between the generation of the head flit to the delivery of the tail flit to the destination. The average latency is computed by dividing the sum of the latencies of each packet over the simulation run by the total number of flits sent. The SystemC model of the NoC implements wormhole routing and weighted round robin packet scheduling. Moreover, each packet has one header flit, one tail flit, and four payload flits.

### A. Impact of the Application Characteristics

The SoC applications used in this experiment were: a Multi-Window Displayer (MWD), an MPEG4 decoder (MPEG4), a Video Object Plane Decoder (VOPD) as well as two applications, called dVOPD and tVOPD obtained by instantiating two and three VOPDs, respectively sharing a common memory. We assumed a $100nm$ technology and a clock frequency of $1.5GHz$. The link capacity $b_{max}$ was set to $1.12GBps$

| **Name** | $\|V_C\|$ | $\|E_C\|$ | Area ($mm^2$) | Total Bw. ($Gbps$) | Ref. |
|---|---|---|---|---|---|
| MWD | 12 | 13 | $3 \times 4$ | 8.96 | [6] |
| MPEG4 | 12 | 27 | $3 \times 2.35$ | 27.8 | |
| VOPD | 12 | 15 | $1.53 \times 1.18$ | 27.9 | |
| dVOPD | 26 | 34 | $2 \times 2.23$ | 66.6 | [26] |
| tVOPD | 38 | 51 | $2.78 \times 2.37$ | 98.84 | |
| VProc | 42 | 69 | $8 \times 6$ | 78.2 | |

TABLE I
Characteristics of the selected SoCs applications.

We used six libraries of communication components differing for the flit-width of the data path (32 bits and 128 bits corresponding to $280 \cdot 10^6$ and $70 \cdot 10^6$ flits per second, respectively) and the size of the largest switch available in the library ($2 \times 2$, $5 \times 5$ and $8 \times 8$).

The results are reported in Figure 12. Each histogram is divided into five zones, one for each application. Each zone contains six bars, one for each library.

The power consumption and the area occupied by the network are increasing functions of the total bandwidth requirement. Most test cases do not need the instantiation of large routers. For instance, the number of input and output ports on each router in the NoCs supporting the MWD and the VOPD applications is no greater than two since each core is a source and/or destination of few communication constraints. These NoCs are basically a set of dedicated point-to-point links with very little sharing. Hence, the difference between the maximum and the average latency is small.

The dVOPD and tVOPD applications show the effect of merging different communications into a common link. In these applications, a central memory is shared among a few cores. Since the memory has only one input and one output port, one or more routers are needed to merge concurrent accesses to memory using time multiplexing. Allowing the installation of larger routers provides two advantages: (1) the total power consumption is reduced (14% and 12% for dVOPD and tVOPD, respectively) as a consequence of a reduced hop count, and (2) the end-to-end latency (both the maximum and average value) is reduced. The latency decrease is modest compared to the reduced number of hops because the time spent for contention among the input FIFOs grows with the router size. Generally, however, for these applications the reduced number of hops counterbalances the negative effect due to contention.

In the MPEG4 application, the SDRAM is shared among many more cores than in the case of the dVOPD and tVOPD applications. Hence, to use larger routers do not give the same benefits. Despite the significant differences between the maximum number of hops in the $2\times2$ and $8\times8$ cases, there are no gains in terms of maximum latency, which in fact is even worse for larger routers (a 73% increase with respect to the $2 \times 2$ case). Here, port contentions cannot be counterbalanced by the reduced hop count, as opposed to the dVOPD and tVOPD cases where routers have no more than five inputs.

This set of test cases shows that the power consumption and the area occupied by the NoC implemented with 32-bit links is much smaller than in the 128-bit implementation. The latter case gives smaller link utilization (i.e. flit rate), which reduces the latency due to contention. This, however, is a minor gain

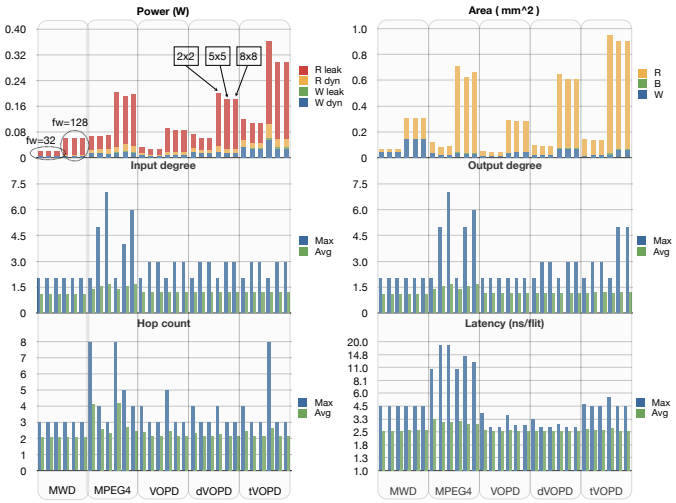


Fig. 12. Properties of the synthesized NoCs for the MWD, MPEG4, VOPD, dVOPD and tVOPD applications. Power is expressed in $Watts$, area in $mm^2$ and latency in $ns/flit$. We used the following notation: `R` for routers, `W` for wires, `B` for sequential buffers. Latency is reported on a logarithmic scale.

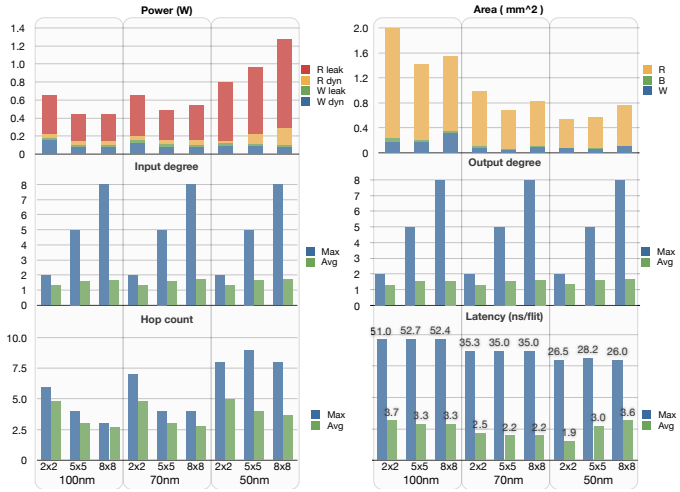


Fig. 13. Properties of the synthesized NoCs for the VProc applications.

and does not justify the use of wider data parallelism, which should be limited to cases when the required bandwidth cannot be achieved with narrower links.

### B. Effect of Technology Scaling

For the second set of experiments we selected the VProc SoC as a representative embedded system application and we studied the impact of scaling the technology on the performance and cost of the synthesized NoC. VProc features a central memory that serves 25 different cores. Each core requires a write and read bandwidth of $960Mbps$ (in each direction). Technology scaling generally enables higher transistor densities and clock frequencies. Hence, as we scale the technology we double the bandwidth requirements from each core to the central memory while keeping the core size fixed. This choice mimics the fact that two cores can fit in the area of one, as the transistor density doubles with the new process generation.

We used a total of 9 libraries obtained as the combination of three different technology processes with three different router designs: specifically, we used $100nm$, $70nm$, and $50nm$ technologies while the routers' maximum size was set equal to $2\times2$, $5\times5$ and $8\times8$, respectively. The clock frequency was set to $1.5Ghz$ at $100nm$. Since the total memory bandwidth is $3GBps$, we set the flit width to 128 bits to achieve a link capacity of $3.2GBps$ with a maximum flit rate of $200\cdot10^6$ per input port of the routers. We increased the clock frequency to $2.25GHz$ and $3GHz$ for the 70 and $50nm$, respectively. We also increased the link capacities to 6.4 and $12.8GBps$ for the two technologies, respectively. The synthesis constraints were set as follows: the density of the installation sites was fixed to 20, which gives a total of 364 different possible locations to install the routers; the synthesis goal is minimum latency with a constraint that each path be no longer than 10 hops. The results are reported in Figure 13.

The critical sequential length drops from $9.98mm$ at $100nm$ down to $3.47mm$ at $50nm$ due to the different electrical parameters of the wires and also to the increased clock frequency (from 1.5 to 3 $GHz$). Since the chip size is $8\times6mm^2$, the entire chip can be spanned in one clock cycle at $100nm$ while 3 cycles are needed at $50nm$. Hence, it is not surprising that the maximum number of hops at $50nm$ does not change much across the various router configurations because intermediate FIFOs are needed to segment long interconnection links. The use of larger routers does not help to reduce the number of hops while it increases the chance of contention. Therefore, contrary to the $100nm$ case, the reduced number of hops does not balance the higher contention probability and, ultimately, the average latency obtained by simulation actually increases.

In terms of power consumption, while there is an advantage in using larger routers with the $100nm$ technology, the picture changes at $50nm$. In fact, the higher clock frequency and the increase in the transistor leakage power discourage the use of larger routers. When stateful repeaters are needed to span large distances among the cores, it is more efficient to spatially distribute small routers on the chip. Finally, the result highlights the need for more accurate timing models for synthesis rather than the simple measure based on hop count.

### C. Quality of the Solution

Ideally, we would like to compare the *exact* solution of problem `PR2` with the one found by our heuristic algorithm. However, the best we can do is to compare our results with a lower bound since even the relaxed ILP algorithm sketched in Section V-C has prohibitive running times for our test cases. We relaxed the problem further by converting the ILP into a Linear Program (LP). We used CPLEX [27] to solve the linear program. We then computed the ratio of the power consumption of the solution found by CPLEX over the one of our heuristic algorithm across various benchmarks. Table II report these results together with the number of positions $|D_{\{xy\}}|$, the computation time '$t_{cpu}$ LP' of CPLEX and the computation time '$t_{cpu}$ H' of our heuristic.

In most cases our heuristic algorithm is 2-3 orders of magnitude faster than solving the LP (a remarkable fact since the LP DOES not find a feasible solution). The power of the NoC found by the heuristic is within 2x from the power found by CPLEX that is very optimistic for the change in the cost

| Name | $\|\hat{D_{xy}}\|$ | $t_{cpu}$ LP | $t_{cpu}$ H | Ratio |
|---|---|---|---|---|
| MWD 2x2 | 94 | 4.76 | 0.11 | 1 |
| MWD 5x5 | 94 | 4.83 | 0.11 | 1 |
| MWD 8x8 | 94 | 4.78 | 0.11 | 1 |
| MPEG4 2x2 | 117 | 434 | 5.29 | 0.49 |
| MPEG4 5x5 | 117 | 479 | 1.46 | 0.55 |
| MPEG4 8x8 | 117 | 394 | 1.32 | 0.48 |
| VOPD 2x2 | 63 | 1.98 | 0.13 | 0.73 |
| VOPD 5x5 | 63 | 0.87 | 0.13 | 0.78 |
| VOPD 8x8 | 63 | 0.85 | 0.13 | 0.78 |
| dVOPD 2x2 | 147 | 130 | 1.8 | 0.69 |
| dVOPD 5x5 | 147 | 60 | 1.66 | 0.66 |
| dVOPD 8x8 | 147 | 60 | 1.65 | 0.66 |
| tVOPD 2x2 | 150 | 438 | 4.54 | 0.71 |
| tVOPD 5x5 | 150 | 423 | 3.32 | 0.66 |
| tVOPD 8x8 | 150 | 426 | 3.34 | 0.66 |

TABLE II

EVALUATING THE HEURISTIC ALGORITHM OF FIGURE 10.

function and for the relaxation of the integer constraints.

## VII. CONCLUSIONS AND FUTURE WORK

We presented a design methodology with a supporting tool infrastructure that follows the Platform-Based Design paradigm and relies on a solid mathematical foundation to model and compose communication networks. We showed how a communication problem can be specified as a point-to-point network and how specifications can be composed, thus enabling an incremental design methodology. The same mathematical formalism is used to model the *platform* that supports on-chip communication (OCC) design. The platform captures all possible communication structures that can be built by assembling the components from the target communication *library*. We formulated a general optimization problem for OCC synthesis that applies to a wide class of libraries. We introduced COSI-OCC, an open software infrastructure for OCC design, and we proposed an efficient heuristic to solve the NoC design problem. Finally, we presented two sets of experiments. The experiments show that little is gained in using complex routers for OCC. Routers should be kept as simple as possible according to the application needs. For memory-centric applications, we showed how as technology scales, to distribute many small routers on chip yields a better communication implementation than using few high-degree routers.

We believe that our software infrastructure together with the formal definitions of specification, library, platform, and abstraction levels, can serve as a "playground" for researchers to compare different algorithmic strategies for the optimization of OCC. We plan to continue the development of COSI-OCC by increasing the set of available components to build a richer platform together with composition rules that ensure the exploration of a larger design space. We also plan to develop accurate models of the data traffic among cores, particularly source burstiness, and to characterize the router latency so that we can account for the real latency of packets traversing the OCC structure rather than using the less realistic metric based on hop counts. Extension to general communication design involving distributed embedded systems such as automobiles and intelligent buildings will be reported shortly.

## REFERENCES


[1] J. D. Meindl, "Interconnect opportunites for gigascale integration," *IEEE Micro*, 2003.

[2] OCP-IP. [Online]. Available: http://www.ocpip.org/home

[3] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Vberg, M. Millberg, and D. Lindqvist, "Network on chip: An architecture for billion transistor era," in *Proc. of the IEEE NorChip Conference*, Nov. 2000.

[4] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. of the Design Automation Conf.*, June 2001.

[5] L. Benini and G. D. Micheli, "Networks on chip: A new SoC paradigm," *IEEE Computer*, 2002.

[6] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. D. Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, Feb. 2005.

[7] S. Murali and G. D. Micheli, "SUNMAP: A tool for automatic topology selection and generation for NOCs," in *Proc. of the Design Automation Conf.*, June 2004, pp. 914–919.

[8] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, Nov. 2005.

[9] K. Lahiri, A. Raghunathan, and S. Dey, "Design space exploration for optimizing on-chip communication architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 23, no. 6, pp. 952–961, Dec. 2004.

[10] U. Ogras and R. Marculescu, "Application-specific network-on-chip architecture customization via long-range link insertion," in *Proc. Intl. Conf. on Computer-Aided Design*, Nov. 2005.

[11] S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. D. Micheli, and L. Raffo, "Designing application-specific networks on chips with floorplan information," in *Proc. Intl. Conf. on Computer-Aided Design*, Nov. 2006, pp. 355–362.

[12] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear-programming-based techniques for synthesis of network-on-chip architectures," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 4, pp. 407–420, Apr. 2006.

[13] ——, "Application specific network-on-chip design with guaranteed quality approximation algorithms." in *ASPDAC*, January 2006.

[14] A. Sangiovanni-Vincentelli, "Defining platform-based design," *EEDesign of EETimes*, February 2002.

[15] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning : Enabling hierarchical design," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 1120–1135, December 2003.

[16] *ISO/IEC 7498-1, Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*, 1994.

[17] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. 36, no. 5, pp. 547–553, 1987.

[18] Y. Hu, H. Chen, Y. Zhu, A. A. Chien, and C.-K. Cheng, "Physical synthesis of energy-efficient networks-on-chip through topology exploration and wire style optimizationz." in *ICCD*, 2005, pp. 111–118.

[19] "The communication synthesis infrastructure (COSI)." [Online]. Available: http://embedded.eecs.berkeley.edu/cosi/

[20] H. S. Wang, X. Zhu, L. S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *Proc. of the 35th Intl. Symp. on Microarchitecture*, Nov. 2002, pp. 294–305.

[21] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Reading,MA: Addison-Wesley, 1990.

[22] S. Heo and K. Asanovic, "Replacing global wires with an on-chip network: a power analysis," in *Proc. of the Intl. Symp. on Low Power Electronics and Design*, 2005, pp. 369–374.

[23] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proceedings of the IEEE*, pp. 490–504, April 2001.

[24] G. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, 1980.

[25] M. Desrochers and F. Soumis, "A generalized permanent labelling algorithm for the shortest path problem with time windows," *Information Systems and Operations Research*, vol. 26, no. 3, pp. 191–212, 1988.

[26] A. Pullini, F. Angiolini, P. Meloni, D. Atienza, S. Murali, L. Raffo, G. D. Micheli, and L. Benini, "65 nm NoC design: Opportunities and challenges," *Proc. of the 1st Intl. Symp. on Networks-on-Chips*, 2007.

[27] "CPLEX." [Online]. Available: http://www.ilog.com/products/cplex/