Constraint-Driven Communication Synthesis

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

Department of Electrical Engineering and Computer Science
University of California at Berkeley, Berkeley, CA 94720-1772
E-mail: {apinto,lcarloni,alberto}@ic.eecs.berkeley.edu

15th March 2002

Abstract

Constraint-driven Communication Synthesis enables the automatic design of the communication architecture of a complex system from a library of pre-defined Intellectual Property (IP) components. The key communication parameters that govern all the point-to-point interactions among system modules are captured as a set of arc constraints *in the* communication constraint graph. *Similarly, the communication features offered by each of the components available in the IP communication library are captured as a set of* feature resources *together with its cost figures. Then, every communication architecture that can be built using the available components while satisfying all constraints is implicitly considered (as an* implementation graph *matching the constraint graph) to derive the optimum design solution with respect to the desired cost figure. The corresponding constrained optimization problem is efficiently solved by a novel algorithm that is presented here together with its rigorous theoretical foundations.*

1 Introduction

In this work we propose a novel approach to design the communication architecture for a system of computational modules whose interaction is specified from an abstract point of view as a collection of communication requirements on a set of point-to-point uni-directional "virtual" channels. By abstracting away the specific functionality of each module, we can focus on exploring the various communication topologies that can be built composing a set of library elements that include "passive elements" (links) as well as active ones (repeaters, switches), each of them coming with a fixed cost function that captures an application-specific optimality criterion. The proposed approach lies on top of a mathematical model that allows us to fully separate computational issues from communication ones. While each computational module acts on the data streams that travel within the system (reading from input channels and writing new data onto the output channels according to its functionality) the communication elements limit themselves to transfer the data between two points (the links), two receive and re-transmit the same data (repeaters) or to route in the proper direction (the switches). This clear task division allows us to focus on the complete exploration of all the possible communication architecture topologies that can be built composing these primitive building blocks.

In fact, we limit ourselves to three main composition types to physically implement the virtual channels: the segmentation of long channels by inserting repeaters between shorter links, the duplication of bandwitdh-challenged channels by adding extra parallel links together with a pair of mux/demux switches, and the merging of distinct channels (which generally involves segmentation and duplication). Different from previous approaches to the problem of communication synthesis, we rely on the definition of a fine-grain library whose elements are combined to derive a communication topology that tightly match the system structure. The proposed algorithm discards all the sub-optimal local solutions, while generating a core set of candidate channel implementations from which it picks the optimum-cost subset based on the library cost functions. By composing this subset the algorithm returns the detailed topology of the final optimum-cost architecture that is guaranteed to satisfy all the original requirements.

1.1 Related Work

Among the several releted papers published in recent years, the authors of [4] split the development of the communication architecture in two steps: channel binding and channel mapping. The former binds virtual communication units to high-level communication channels, while the latter associates to each unit a tree of alternative physical implementations from a library. Then a depth-first search strategy is used to derive an optimal solution. In [8] and [9] the design of the communication architecture is done with an exploration of different solutions validated by a fast performance simulation that is based on a detailed characterization of the library components The focus of [13] is on interface synthesis among processing elements that communicate on a bus-based architecture. The authors of [11] assume that the network topology is given and find an efficient physical implementation that allows to achieve very high performance by

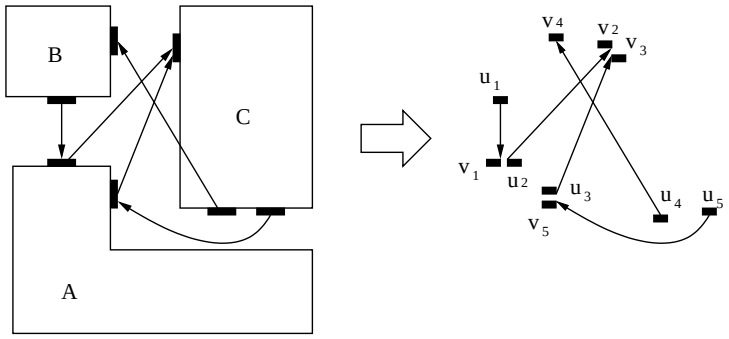

Figure 1: Abstract model of communication requirements.

sending control signals and deadlines on the delivery of the message in advance to the corresponding data. The approach of [2] is similar to the one of the present work altough specialized to ATM networks: the problem is to select the topology of a network composed of links specified in a library with their speed and cost. Different from our approach this paper assumes that the location of the intermediate communication nodes is fixed and the optimization is limited to link selection.

2 Communication Constraint Graphs

The abstract model to specify a communication system is represented in Figure 1 and consists of a set of computational modules communicating through point-to-point unidirectional communication "virtual" channels that are connected to the modules by means of input/output ports. A module may communicate with another module through multiple unidirectional channels (in both directions). For each entering (leaving) channel connected to the module there is a corresponding dedicated input (output) port. Communication requirements are specified for each channel as a set of two parameters: the distance to be covered and the required bandwidth. Our intent is to use this model as a basic common starting point to define the communication specifications of various kinds of systems, such as a "System-on-Chip", a multi-chip multi-processor system, or a local area network (LAN). Naturally, the basic model will be appropriately extended/refined for each particular application. For instance, in case of a "System-on-Chip", a channel could represent the set of wires implementing the address bus that a processor uses to access a cache memory and a certain required channel bandwidth could be specified in gigabyte per second. Furthermore, for each port of every computational module on the chip a certain location could be specified, thus making it possible to compute the Manhattan distance between any two communicating ports. On the other hand, if we are studying how to implement a LAN and we want to evaluate whether to realize it as a fiber-optic network or a wireless network, (or a combination of the two), the set of channels could just capture all the specified links among the clients and the servers. Here the Euclidean distance among all these components could be sufficient, while for each channel, the bandwidth is usually specified in gigabit per second.

Independently from the specific application, we follow the principle of orthogonalization of concerns [1, 7] and we uniquely derive from the network a *communication constraint graph* that allows us to focus on the design of the communication architecture while disregarding the functionality of each computational module in the system. Working with the constraint graph, we define the *constraint-driven communication synthesis problem* as the task of finding the communication architecture which satisfies all the constraints specified as communication requirements on the channels, while minimizing a predefined cost function that captures an optimality criterion which must be defined for the specific application.

Definition 2.1 *A* communication constraint graph, *or simply* constraint graph, *$\mathcal{G} = G(V,A)$ is a directed graph, where each vertex v is associated to a port of computational module of the system and each directed arc a (also called, simply,* constraint arc*) represents a point-to-point communication channel between two modules. A position $p(v)$ is assigned to each vertex $v \in V$, while the following quantities are associated to each directed arc $a = (u,v)$ and referred to as* arc properties*:*

- *$d(a)$: arc* length *(or,* distance*) between vertex u and vertex v.*
- *$b(a)$: communication bandwidth on the constrained arc a.*

The right-end side of Figure 1 illustrates the communication constraint graph that is derived from the network on the left-end side. The previous definition doesn't specify whether the position of the vertices should be considered on the plane or in space, nor which type of distance is used to compute the arc length. However, for all arcs $a = (u,v)$ in the graph, the values of $d(a)$ must be consistent with the positions $p(u)$ and $p(v)$. For instance, in the case of a "System-on-Chip", the position of vertex v (corresponding to a module port) will be given by its coordinates $p(v) = (x_v, y_v)$ and the length of the arc $a = (u,v)$ may be computed using the Manhattan distance between the coordinates of its two nodes $a = |x_u - x_v| + |y_u - y_v|$. In the sequel, we will rely on the notion of geometric norm $||p(u) - p(v)||$ to identify the generic distance between two vertices $u, v \in \mathcal{G}$.

The set of all arc properties (lengths and bandwidths) represent the set of design constraints that need to be satisfied while deriving a communication architecture that implements all point-to-point communication channels of the system. As discussed in the introduction, we assume that this communication architecture is realized by putting together elements taken from a communication library. In particular, the library may contain several kinds of communication links, repeaters, and switches. For instance, a communication link guarantees that a certain flow of information can be transferred with up to a specified bandwidth between two ports as long as they lie within a specified distance. Examples of communication links are optical fiber connections, wireless links, or metal lines on a chip that can sustain up to a certain bandwidth given a certain distance. A repeater is used to connect to links (that are able to sustain a certain bandwidth) to cover a distance that they would not to be able to cover stand-alone. A switch, while being able to act as a repeater, enables the connection of multiple links that share a specified bandwidth. A multiplexer is a switch that takes multiple incoming links and "merges" them into one outgoing link whose bandwidth is larger than the sum of the incoming one. A de-multiplexer does the inverse function. In the sequel we define more formally the notion of communication library and we show how putting together these basic elements and defining a few simple operation to combine them we are able to build a rich set of heterogeneous communication architectures having various topologies and bandwidth characteristics.

Definition 2.2 *A* communication library $\mathcal{L} = L \cup N$ *is a collection of communication links and communication nodes. Each link* $l \in L$ *in the library is characterized by a set of* link properties*:*

- $d(l)$ *: the link* length *(or,* distance*) corresponds to the length of the longest communication channel that can be realized by this link.*
- $b(l)$ *: the link* bandwidth *corresponds to the bandwidth of the fastest communication channel that can be realized by this link.*
- $c(l)$ *: the link* cost *is a figure (the lower the better) defined with respect to the other links in the library based on an optimality criterion that may vary with the type of application.*

Also, each communication node $n \in L$ *has a cost* $c(n)$.

The realization of a communication architecture that satisfies the requirements specified by a constraint graph can be modeled as a set of graph transformations (including the addition of new arcs and vertices). This leads us to define a new graph, called implementation graph whose set of vertices is an extension of the set of vertices of the constraint graph. In particular, each vertex in the implementation graph is either a "computational vertex" (corresponding to a vertex in the original constraint, i.e. a port of a computational module of the original system) or a newly added "communication vertex", corresponding to an instance of a communication node from the library. Also, every arc in the graph is mapped to a library link.

Definition 2.3 *A path* $q = (v_1, a_1, v_2, a_2, \ldots, v_{Q-1}, a_{Q-1}, v_Q)$ *of a graph* $G = G(V,A)$ *is an alternating sequence of distinct vertices and arcs in G, with* $V(q,G)$ *and* $A(q,G)$ *denoting respectively the set of vertices and arcs touched by q. Furthermore, we define the* sub-path *of p up to vertex* $v_j \in V(q,G)$ *as* $sub(q,v_j) = (v_1, a_1, v_2, a_2, \ldots, a_{j-1}, v_j)$*. As for an arc we can define the following* path properties*:*

- *(length):* $d(q) = \sum_{i=0}^{Q-1} d(a_i)$.
- *(bandwidth):* $b(q) = \min_{(i=0,\ldots,Q-1)}(b(a_i))$.
- *(cost):* $c(q) = \sum_{i=0}^{Q-1} c(a_i)$.

where $c(a_i)$ denotes the cost of an arc as it specified in the following definition.

Definition 2.4 *Given a constraint graph* $\mathcal{G} = G(V,A)$ *and a communication library* $\mathcal{L} = L \cup N$*, an* implementation graph $\mathcal{G}'(\mathcal{G},\mathcal{L}) = G(V' \cup N', A')$ *is a directed (possibly multi-)graph s.t.:*

- *for each vertex in V there is a corresponding vertex in* V' *and vice versa (and they have the same positions), i.e. there is a bijective mapping function* $\chi : V \to V'$ *s.t.* $\forall v \in V, \exists v' \in V'(v' = \chi(v) \wedge v = \chi^{-1}(v') \wedge p(v) = p(v'))$.
- *for each vertex in* N' *there is a corresponding communication node in N, i.e. there is a surjective mapping function* $\psi : N' \to N$ *s.t.* $\forall n' \in N', \exists n \in N(n = \psi(n'))$.
- *for each arc in* A' *there is a corresponding communication link in L and they share the values of their properties, i.e. there is a surjective mapping function* $\phi : A' \to L$ *s.t.* $\forall a' \in A', \exists l \in L(l = \phi(a') \wedge d(a') = d(l) \wedge b(a') = b(l) \wedge c(a') = c(l))$.
- *for each arc* $a = (u,v)$ *in the constraint graph there is a set of paths* $\mathcal{P}(a)$ *in the implementation graph connecting* $\chi(u)$ *to* $\chi(v)$ *without passing through any other computational vertex (but only, possibly, through communication vertices) that together satisfy the bandwidth constraint* $b(a)$ *as the sum of the bandwidth* $b(q)$ *of each path* $q \in \mathcal{P}$*. Formally,* $\forall a = (u,v) \in \mathcal{G}, \exists \mathcal{P}(a) \in \mathcal{G}'$ *s.t.* $\forall q = (n_1, \ldots, n_Q) \in \mathcal{P}$*:*

  1. $n_1 = \chi(u) \wedge n_Q = \chi(v) \wedge \forall m \in [2, Q-1](n_m \in N')$.

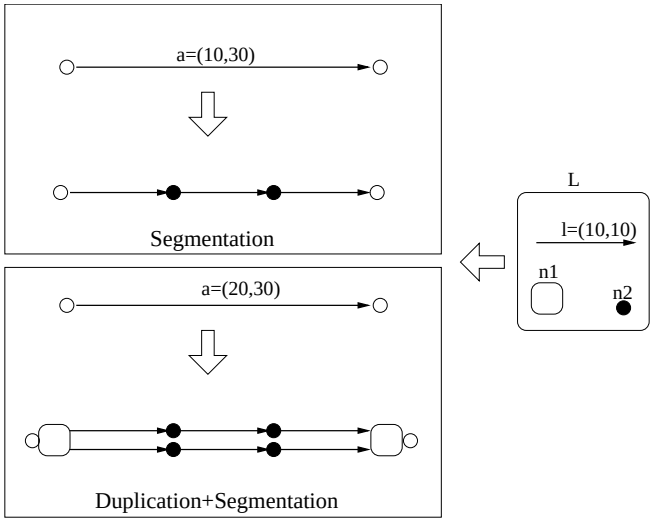

Figure 2: Examples of arc segmentation and duplication.

2. $b(a) \leq \sum_{q \in \mathcal{P}} b(q)$.

The set of paths $\mathcal{P}(a)$ is called the constraint arc implementation *(or, simply,* arc implementation*) and its cost is $C(\mathcal{P}(a)) = \sum_{q \in \mathcal{P}} c(q)$.*

Definition 2.5 *The* cost *of an implementation graph $\mathcal{G}'$ is defined as* [1]*:*

$$C(\mathcal{G}') = \sum_{n' \in N'} c(n') + \sum_{a' \in A'} c(a') \tag{1}$$

where $c(n') = c(\psi(n'))$ and $c(n')$ are as of definition 2.4.

Generally, for a given library there are many possible implementation graphs that satisfy the requirements expressed by the constraint graph while having different costs. In particular, one implementation graph, called the optimum point-to-point implementation graph, is guaranteed to exist and it is derived by implementing a single arc constraint independently from all the others present in the constraint graph.

Definition 2.6 *Given a constraint graph $\mathcal{G} = G(V,A)$ and a communication library $\mathcal{L} = L \cup N$, a* optimum point-to-point implementation graph *$\mathcal{G}'(\mathcal{G},\mathcal{L}) = G(V' \cup N', A')$ is an implementation graph such that $\forall a_i \in G, \mathcal{P}(a_i)$ has the minimum cost $C(\mathcal{P}(a_i))$ while being subject to the constraint that:*

$$\bigcap_{a_i \in G} \mathcal{P}(a_i) = \emptyset,$$

i.e. its arc implementations are disjoint.

The following definition gives a characterization of all possible structures for the arc implementations in an optimum point-to-point implementation graph (see also figure 2).

Definition 2.7 *Given a constraint graph $\mathcal{G} = G(V,A)$ a communication library $\mathcal{L} = L \cup N$, and an* implementation graph *$\mathcal{G}'(\mathcal{G},\mathcal{L}) = G(V' \cup N', A')$ the arc implementation $\mathcal{P}(a)$ of $a(u,v) \in A$ is called:*

- *an* arc matching iff $\mathcal{P}(a) = \{p = (\chi(u), \chi(v))\}$*, i.e. the implementation corresponds to exactly one library link.*
- *a* K-way arc segmentation iff $\mathcal{P}(a) = \{p = (\chi(u), n_1, \ldots, n_{K-1}, \chi(v))\}, \forall k \in [1, K-1](n_k \in N')$*, i.e. the implementation corresponds to the concatenation of K library links.*
- *a* K-way arc duplication iff $\mathcal{P}(a) = \{p_1 = (\chi(u), \chi(v)), \ldots, p_K = (\chi(u), \chi(v))\}$*, i.e. the implementation corresponds to placing K library links in parallel.*

Clearly, the optimum point-to-point implementation graph can be seen as the representation of a communication architecture that is built considering sequentially the implementation of each constraint arc *a* as a stand-alone task, performed according to the following steps:

[1] Note that the "computational vertices" in V are not part of the cost equation, they may be though as having null cost.

1. if it exists, the minimum cost link l in the library that satisfies the constraints $d(l) \geq d(a) \wedge b(l) \geq b(a)$; if such a link exists an arc matching is the desired arc implementation [2].

2. if $d(l) < d(a)$ for all library links l (while $b(l) \geq b(a)$ is satisfied by some l), then arc segmentation will lead to an implementation.

3. conversely, if $b(l) < b(a)$ for all library links l (while $d(l) \geq d(a)$ for some l), then arc duplication will lead to an implementation.

4. in case both constraints can not be satisfied by any link in the library, then a combination of arc segmentation and arc duplication will lead to an implementation..

Lemma 2.1 *For all constraint graphs $\mathcal{G} = G(V,A)$ and all communication libraries $\mathcal{L} = L \cup N$, there exists an* optimum point-to-point implementation graph $\mathcal{G}'(\mathcal{G},\mathcal{L}) = G(V' \cup N', A')$ *and* $C(\mathcal{G}') = \sum_{n' \in N'} c(n') + \sum_{a' \in A'} c(a') = \sum_{a \in A} C(\mathcal{P}(a))$.

On the other hand, by analyzing the definition of implementation graph it is clear that some of its arc implementations may share paths (i.e. links and/or communication nodes). In fact, in general the cost of an implementation graph is smaller than the sum of the costs of its arc implementations, i.e., re-considering equation 1, we have:

$$C(\mathcal{G}') = \sum_{n' \in N'} c(n') + \sum_{a' \in A'} c(a') \leq \sum_{a \in A} C(\mathcal{P}(a)) \tag{2}$$

As a consequence, we are forced to analyze the interactions between point-to-point constraint arc implementations and the task of finding the optimum implementation graph becomes more challenging.

Definition 2.8 *Given a constraint graph $\mathcal{G} = G(V,A)$ a communication library $\mathcal{L} = L \cup N$, and an* implementation graph $\mathcal{G}'(\mathcal{G},\mathcal{L}) = G(V' \cup N', A')$, *the union of $K \in [2, |A|]$ arc implementations $\mathcal{P}(a_1), \ldots, \mathcal{P}(a_K)$ is called a K-*way arc merging *when $\exists q^\star$ s.t. $\bigcap_{k=1}^{K} \mathcal{P}(a_k) = q^\star$. The path $q^\star$ is called the* common path *of the merging transformation.*

It becomes natural to define a constrained optimization problem aimed to find that implementation graph whose cost (expressed as the sum of the cost of all its components mapped to a library element) is minimum.

Problem 2.1

Given: a constraint graph $\mathcal{G} = G(V,A)$ and a communication library $\mathcal{L} = L \cup N$

Minimize: the cost $C(\mathcal{G}')$

Over all: *implementation graphs* $\mathcal{G}'(\mathcal{G},\mathcal{L}) = (V' \cup N', A')$ of $\mathcal{G}$.

Clearly, this problem can be seen as a special case of 0-1 integer linear programming (ILP).

In the sequel, we will present an exact algorithm to find the solution of this constrained optimization problem when the following assumption holds:

Assumption 2.1 *Given a constraint graph $\mathcal{G} = G(V,A)$ and a communication library $\mathcal{L} = L \cup N$, for each constraint arc $a = (u,v)$, $C(\mathcal{P}(a)) > 0$ and for all pairs of arcs $a = (u,v), a' = (u',v') \in A$ and for all corresponding minimum-cost constraint arc implementations $\mathcal{P}(a), \mathcal{P}(a')$:*

$$((d(a) \leq d(a') \wedge b(a) \leq b(a')) \Leftrightarrow (C(\mathcal{P}(a)) \leq C(\mathcal{P}(a')))) \tag{3}$$

This assumption is justified from a practical point in most application domains: for instance an optical fiber supporting a given bandwitdh is priced per meter; similarly, for radio link covering a fixed distance the higher is the desired bandwith the more expensive is the cost of the equipment.

3 Solving the Constrained Optimization Problem

To solve exactly the constrained optimization problem defined in the previous section we developed an algorithm that is based on a sequence of two steps, namely *local solution generation* and *global solution derivation*:

1. We efficiently generate the set $\mathcal{S}$ of all those alternative distinct implementations of each arc in the constraint graph that are not "dominated" by other less expensive implementations. The set $\mathcal{S}$ includes all $|A|$ arc implementations in the optimum point-to-point implementation graph (see 2.6) together with a minimal set of arcs implementations that are built applying k-way arc mergings ($k \in [2, |A|]$) (possibly combined with some arc segmentation/duplication). The elements of $\mathcal{S}$ are said "local" since they generally provide an implementation only for a subset of the constraint arcs, and, in the case of $k = |A|$-way mergings (where all arcs are implemented), the implementation may only represent a locally-minimum in the search of the solution space.

[2] See also the assumption 2.1 defined below.

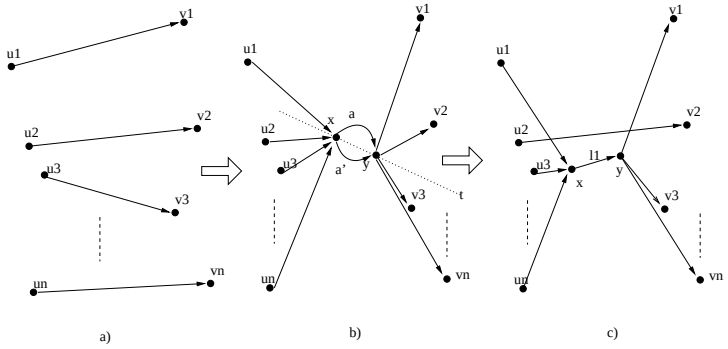

Figure 3: Example of split in a k-way merge

2. After computing the cost of each element of $\mathcal{S}$, we solve an instance of the weighted Unate Covering Problem(UCP) [3], to find that subset of $\mathcal{S}$ providing the minimum cost global implementation for all arcs of the constraing graph.

When combined, the two steps correspond to implicitly considering all possible communication sub-architectures that can be generated by putting togheter communication library components while being compatible with the requirements of the constraint graph. To be effective, this approach must rely on the ability of generating the smaller possible set of local solutions $\mathcal{S}$ that must necessarily be considered to guarantee that the entire solution space is explored as part of the exact search of the global optimum. One could naively think to generate all possible solutions and leave the responsability of finding the global optimum to state-of-the-art UCP solvers [3, 6, 10], which provide sophisticated techniques to prune away suboptimal local solutions. In practice, this would turn out to be quite expensive from a computational point-of-view during both steps. Instead, we will present here a set of theoretical results that guide us during the first step, enabling the pruning of many useless branches during the search of the tightest $\mathcal{S}$.

Since, the proliferation of alternative arc implementations (on top of those of the optimum point-to-point implementation graph) is due to the possibility of realizing K-way arc merging implementations, it is natural to focus on defining criteria that establish when a subset of K arcs can be merged.

Definition 3.1 *Let $\mathcal{G} = G(V,A)$ be a constraint graph and $\mathcal{L} = L \cup N$ a communication library. Also $\forall k \in [2, |A|-1]$, let A^k denote a subset of A having cardinality $|A^k| = k$ and V^k the vertices connected by the arcs of A^k. Let $\mathcal{G}^k = G(V^k, A^k)$ be the projection of $\mathcal{G}$ onto A^k. A^k is said to be* k-way mergeable iff *the union of the arc implementations of the minimum-cost implementation graph $\mathcal{G}^\star(\mathcal{G}^k, \mathcal{L})$ of $\mathcal{G}^k$ is a k-way merging. The set of sets of k-way mergeable arc is denoted as $\mathcal{M}^k$.*

The following lemma provides a sufficient conditions to detect when a pair of arcs is not 2-way mergeable. As for all the remaining results in this section, this condition is valid independently from the characteristics of the chosen communication library $\mathcal{L} = L \cup N$, as long as the library satisfies Assumption 2.1.

Lemma 3.1 *Let $A^2 = \{a, a'\} \subseteq A$ be a subset of two arcs $a = (u,v), a' = (u',v')$ of a constraint graph $\mathcal{G} = G(V,A)$. Then,*

$$\left\{ d(a) + d(a') \leq ||p(u) - p(u')|| + ||p(v) - p(v')|| \right\} \Rightarrow A^2 \notin \mathcal{M}^2$$

Proof. [By contradiction]. Let us assume $A^2 \in \mathcal{M}^2$ and prove that this implies that $||p(u) - p(u')|| + ||p(v) - p(v')|| < d(a) + d(a')$. For any library $\mathcal{L}$, let $\mathcal{G}'(\mathcal{G}^2, \mathcal{L}) = G(V \cup N, A)$ be the graph implementation corresponding to the generic 2-way merging illustrated in figure 4, where $V = \{u, v, u', v'\}$ $N = \{x, y\}$ and $A' = \{(u,x), (y,v), (u',x), (y,v'), (x,y)\}$. Graph $\mathcal{G}'$ has cost:

$$C(\mathcal{G}') = C(\mathcal{P}((u,x))) + C(\mathcal{P}((u',x))) + C(\mathcal{P}((y,v))) + C(\mathcal{P}((y,v'))) + C(\mathcal{P}((x,y)))$$

Instead, let $\mathcal{G}^\star(\mathcal{G}^2, \mathcal{L})$ be the optimum point-to-point implementation graph made of the arc implementations $\mathcal{P}(a), \mathcal{P}(a')$ and cost $C(\mathcal{G}^\star) = C(\mathcal{P}(a)) + C(\mathcal{P}(a'))$. By definition 3.1 of 2-way mergeability, $A^2 \in \mathcal{M}^2 \Rightarrow C(\mathcal{G}') < C(\mathcal{G}^\star)$. Therefore, since $C(\mathcal{P}(x,y)) > 0$, we have that

$$C(\mathcal{P}((u,x))) + C(\mathcal{P}((u',x))) + C(\mathcal{P}((y,v))) + C(\mathcal{P}((y,v'))) < C(\mathcal{P}(a)) + C(\mathcal{P}(a'))$$

Recalling Assumption 2.1 on the library $\mathcal{L}$ and considering that the bandwidth constraints $b(a), b(a')$ must be satisfied by both implementations, the previous inequality on the costs translates on the following inequalities on distances:

$$d((u,x)) + d((u',x)) + d((y,v)) + d((y,v')) < d(a) + d(a')$$

which, by the triangular inequality of the distance, implies that $||p(u) - p(u')|| + ||p(v) - p(v')|| < d(a) + d(a')$. □

The following lemma can be seen as an extension of the result of the previous lemma to the case of k arcs because it provides a sufficient condition that guarantees that they are not k-way mergeable.

[3] This problem can be seen as a matrix formulation of the MINIMUM COVER problem [5].

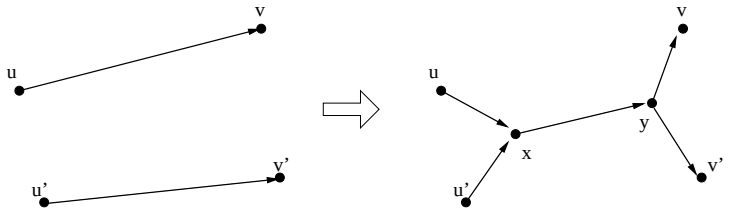

Figure 4: 2-way merge

Lemma 3.2 *Let $A^k = \{a_1, \ldots, a_k\} \subseteq A$ be a subset of k arcs $a_1 = (u_1, v_1), \ldots, a_k = (u_k, v_k)$ of a constraint graph $\mathcal{G} = G(V,A)$. Then,*

$$\left\{\left((k-1)\cdot d(a_k) + \sum_{i=1}^{k-1} d(a_i)\right) \leq \sum_{i=1}^{k-1} ||p(u_i) - p(u_k)|| + ||p(v_i) - p(v_k)||\right\} \Rightarrow A^k \notin \mathcal{M}^k$$

Proof. [By contradiction]. Let us assume $A^k \in \mathcal{M}^k$ and prove that this implies

$$\left((k-1)\cdot d(a_k) + \sum_{i=1}^{k-1} d(a_i)\right) > \sum_{i=1}^{k-1} ||p(u_i) - p(u_k)|| + ||p(v_i) - p(v_k)|| \tag{4}$$

Let $q^\star = (x,y)$ be the *common path* of the *k*-way merging for the arcs of A^k. For each constraint arc $a_i = (u_i, v_i) \in A^k$, a necessary condition to be part of the *k*-way merging is

$$d(a_i) > d((u_i, x)) + d((y, v_i)) \tag{5}$$

Without loss of generality, single out the constraint arc $a_k = (u_k, v_k) \in A^k$. Then, sum both sides of inequality(5) for all other $k-1$ arcs $a_1, \ldots, a_{k-1}$. Finally, sum $k-1$ times to both sides of the resulting inequality the corresponding sides of inequality(5) written for arc a_{k-1}. By applying the triangular inequality to the a_{k-1} pairs of distances $d((u_i,x)) + d((y,v_i))$ with respect to the position of the nodes *u* and *v* we obtain inequality 4. □

Theorem 3.1 *Let $\mathcal{G} = G(V,A)$ be a constraint graph. Then,*

$$\begin{aligned} &\left\{\forall k \in [2, |A|-1], \forall A^{k-1} \subset A \setminus \{a\}, \left(A^{k-1} \cup \{a\} \notin \mathcal{M}^k\right)\right\} \quad \Rightarrow \\ \Rightarrow \qquad &\left\{\forall h \in [k, |A|], \forall A^h \subset A \setminus \{a\} \left(A^h \cup \{a\} \notin \mathcal{M}^{k+h}\right)\right\} \end{aligned}$$

Proof. Let $A^k \subset A \setminus \{a\}$ be an arbitrary set of *k* constrained arcs of $\mathcal{G}$ that doesn't contain $a = (u,v)$ nd let $A^{k+1} = A^k \cup \{a\}$. Let's consider all *k* distinct subsets of A^{k+1} that contains *a*. For for each of these subset, we apply Lemma 3.2 to write the condition that excludes its *k*-way mergeability:

$$\begin{aligned} (k-1)\cdot d(a) + d(a_2) + \cdots + d(a_k) \quad &\leq \\ \leq ||p(u_2) - p(u)|| + \cdots + ||p(u_k) - p(u)|| \quad &+ \\ + ||p(v_2) - p(v)|| + \cdots + ||p(v_k) - p(v)|| & \\ \vdots & \\ (k-1)\cdot d(a) + d(a_1) + \cdots + d(a_{k-1}) \quad &\leq \\ \leq ||p(u_1) - p(u)|| + \cdots + ||p(u_{k-1}) - p(u)|| \quad &+ \\ + ||p(v_1) - p(v)|| + \cdots + ||p(v_{k-1}) - p(v)|| \quad &\Rightarrow \end{aligned}$$

By summing the $k-1$ inequalities is:

$$\begin{aligned} k\cdot(k-1)\cdot d(a) + (k-1)\cdot d(a_1) + \cdots + (k-1)\cdot d(a_k) \quad &\leq \\ \leq (k-1)||p(u_1) - p(u)|| + \cdots + (k-1)||p(u_k) - p(u)|| \quad &+ \\ + (k-1)||p(v_1) - p(v)|| + \cdots + (k-1)||p(v_k) - p(v)|| \quad &\Rightarrow \end{aligned}$$

If we divide both sides of the previous inequality by $k-1$, a quantity greater than zero, and we recall Lemma 3.2, we obtain the condition that guarantees that A^{k+1} is not $k+1$-way mergeable. Due to the arbitrary choice of the set A^k, we have proven that

$$\left\{\forall A^k \subset A \setminus \{a\}, \left(A^k \cup \{a\} \notin \mathcal{M}^k\right)\right\}$$

This corresponds to the thesis for the case $h = k$. The corresponding result for all $h \in [k+1, |A|]$ can be obtained by applying recursively the same procedure . □

Given a constraint graph $\mathcal{G} = G(V,A)$ and communication library $\mathcal{L} = L \cup N$, the following result provides a sufficient condition to establish that a subset A^k of A is not k-way mergeable, i.e., formally, that $A^k \notin \mathcal{M}^k$.

Theorem 3.2 *Let $A^k = \{a_1, \ldots, a_k\} \subseteq A$ be a subset of k arcs $a_1 = (u_1, v_1), \ldots, a_k = (u_k, v_k)$ of a constraint graph $\mathcal{G} = G(V,A)$ and $\mathcal{L} = L \cup N$ be a communication library. Then,*

$$\left\{ \sum_{i=1}^{k} b(u_i, v_i) \geq \left(\max_{l \in L} \{b(l)\} + \min_{j \in [1,k]} \{b(u_j, v_j)\} \right) \right\} \Rightarrow A^k \notin \mathcal{M}^k$$

Proof. Figure 3 helps to understand the proof. By hypothesis, the common path of any possible k-way merging must be decoupled in at least 2 arcs a, a' with $\phi(a) = \{l_1 \in L \mid b(l_1) = \max_{l \in L} \{b(l)\}\}$. Without loss of generality, let a_1 be the arc carrying the minimum bandwidth costraint within A^k, i.e. $a_1 = \min_{j \in [1,k]} \{b(u_j, v_j)\}$. We consider two cases:

1. Assume that the inequality is in fact an equation. This means that we can separate the implementation along the dotted line t, as shown in figure 3-(b), and derive an implementation composed by a path from a_1 source to a_1 destination that includes a', together with a $k-1$-way merge for the other constraint arcs. The cost of the implementation of a_1 is greater or equal than its point-to-point minimum-cost implementation, because the latter has necessarily a smaller distance to cover while the bandwidth is the same (recall Assumption 2.1). Therefore the k-way merge has a cost that is greater than the direct implementation of the link with the minimum bandwidth plus a $k-1$-way merge of the other links (Figure 3-(c)).

2. If the hypothesis is an inequality then two situations may occur: the first, trivial, happens when there are no links in the library that match the $b(a_1)$ and this is the same as the case already discussed. Instead, when there are r arcs implementations sharing a', we can ideally separate the merge into two implementations along the dotted line t. Hence, we can divide the arcs into two sets A^r and A^{k-r} s.t. $A^r \cup A^{k-r} = A^k$ and $A^r \cap A^{k-r} = \varnothing$. Again, two situations may happen. If the two sets are respectively r-way mergeable and $(k-r)$-way mergeable, then the implementation of the two sets separately has a cost less or equal to the k-merge of A^k and, by definition, this means that A^k is not k-way mergeable. If instead, at least one of the two sets is not mergeable, say A^r, this means that the point-to-point implementation of its arcs has a lower cost. Therefore the composition of the implementaion of A^k wiht the $(k-r)$-way merge of A^{k-r} and r point-to-point implementations of the arcs of A^r has a cost lower than the k-way merge of A^k. Hence, A^k is not k-way mergeable. □

Figure 5 illustrates the algorithm to generate a minimal set of candidate arc implementations that is based on the above resuls. First, it is convenient to define two distinct symmetric matrices (the Constrained Distance Sum Matrix Γ and the Merging Distance Sum Matrix Δ) to capture key quantities related to each pair of arcs in the constraint graph $\mathcal{G} = G(V,A)$. In particular, for any two arcs $a_i = (u_i, v_i), a_j = (u_j, v_j)$, $\Gamma(a_i, a_j) = d(a_i) + d(a_j)$ and $\Delta(a_i, a_j) = ||p(u) - p(u')|| + ||p(v) - p(v')||$. Notice that since the two matrices are symmetric, we only need to scan the values of their upper diagonal part.

After having saved into $\mathcal{S}$ the optimum point-to-point arc implementation associated to each constraint arc (loop 3-6), the algorithm proceeds by subsequently considering all possible k-way mergings for incrementing value of k (loop 13-32). Using the result of Theorem 3.1, as soon as no k-way mergings are possible for an arc a_j, the corresponding column (and row) is removed from the matrix Γ (line 22). The algorithm leaves the main loop, when the set of columns of Γ becomes empty (line 27). In general, for any given k, we iterate through the current set of columns of Γ (recall that each of them is associated to an arc that can still be part of a k-way merging) and, for each column a_j, we considerate all possible sets of rows I (also associated to arcs) of cardinality k togheter with a_j as a potentially k-way mergeable subset A^k. If the pruning condition of Lemma 3.2 (line 16) is not satisfied as well as the one of Theorem 3.2, as soon as (line 22) is reached, we are forced to include a possible k-way merging of A^k within $\mathcal{S}$. Instead, if this is not the case for all A^k, we are allowed to avoid considering a_j again. The algorithm terminates returning the set $\mathcal{S}$ of candidate arc implementations, whose exact structure (i.e the exact topology, communication node position, number of links,.../) is later obtained solving a simple linear optimization problem, which computes also the cost. Finally, a unate covering matrix is built by associating to each row a constraint arc, to each column a candidate implementation and setting each entry i, j to one if the implementation j implements the arc i, to zero otherwise. Each columns has also a weigth corresponding to the implementation cost. Finally, the selection of the optimum global solution correspond to the solution of this instance of weighted Unate Covering Problem (UCP) and can be found by using state-of-the-art UCP solvers [3, 6, 10].

4 Domain Application Examples

The algorithm presented in the previous section is illustrated here by means of two examples that are taken from different application fields. The first example represents a simple wide-area network (WAN), while the second example shows how the current approach can be adapted to attack the on-chip communication synthesis problem,

```
1: GenerateCandidateArcImplementations (G, L)
2: {Compute set S of candidate arc implementations}
3: S ⇐ ∅
4: {Get optimum point-to-point arc implementations}
5: for all arc a ∈ G do
6:    P(a) ⇐ findBestPointToPointImplementation (a, L)
7:    S ⇐ S ∪ σ(P(a))
8: end for
9: {Find non-dominated candidate k-way mergings}
10: B ⇐ ComputeBandwidthVector (G)
11: Γ ⇐ ComputeConstrainedDistanceSumMatrix (G)
12: Δ ⇐ ComputeMergingDistanceSumMatrix (G)
13: k ⇐ 1
14: foundCanditateMapping ⇐ TRUE
15: while (foundCanditateMapping) do
16:    for all column j ∈ col(Γ) do
17:       for all row subsets I = {i_1, ..., i_k} ⊆ row(Γ) do
18:          if Σ_{i=1}^{k} Δ[i, j] < Σ_{i=1}^{k} Γ[i, j] then
19:             b_min ⇐ min {B[j], min_{i∈[1,k]} {B[i]}}
20:             if ∃l ∈ L s.t. (b(l) + b_min) < Σ_{i=1}^{k} B[i] + B[j] then
21:                S ⇐ S ∪ findKMergingImplementation(j, I)
22:                foundCanditateMapping ⇐ TRUE
23:             else
24:                col(Γ) ⇐ col(Γ) \ {j}
25:             end if
26:          end if
27:       end for
28:    end for
29:    if col(Γ) = ∅ then
30:       foundCanditateMapping ⇐ FALSE
31:    else
32:       k ⇐ k + 1
33:    end if
34: end while
```

Figure 5: Algorithm to generate all candidate arc implementations.

Figure 6-(a) reports the diagram of a wide-area communication network, where the length of the arcs suggests that the"computational" nodes A,B,C are fairly close to each other as well as nodes D and E, while the two groups are separated by a distance which is relatively much larger. We assume that every channel presents the same bandwidth requirement, namely $10Mbps$. Figure 6-(b) shows the corresponding communication constraint graph where only the ports and the channels are retained. In this case, it is reasonable to adopt the approximation that all the ports of a computation node have the same position. The library that is available for implementing the communication architecture consists of two types of links, whose cost is a function of the supported channel length: a radio link $l_r = (11Mbps, l, \$2 \times meter)$, and an optical link $l_o = (1Gbps, l, \$4 \times meter)$. Table 1 and Table 2 report respectively the values of the Constrained Distance Sum Matrix Γ and the Merging Distance Sum Matrix Δ expressed in kilometers.

By running the algorithm reported in Figure 5, it is easy to determine that arc a_8 is not mergeable with any other arc and, therefore, will have to be implemented as a minimum-cost point-to-point link. Due to its distance, this links turns out to be the radio link. The algorithm also determines that arc a_7 cannot be involved in any 4-way arc merging (nor, therefore, in any k-way mergings with $k > 4$). Besides the 8 optimum-cost point-to-point implementations, the set $\mathcal{S}$ contains thirteen 2-way, twentyone 3-way, sixteen 4-way, and five 5-way candidate arc mergings. For each candidate implementation the following minimization problem is solved to derive its cost as well as the position of its communication nodes. implementation:

Minimize: the cost $C(x)$
Subject to: $K \cdot x = d$

The K matrix is derived by writing the equation forcing that the sum of the lengths along x and y axes must be equal to the difference in position between source and destination points. Finally, after solving the weighted UCP, we learn that the minimum cost solution is

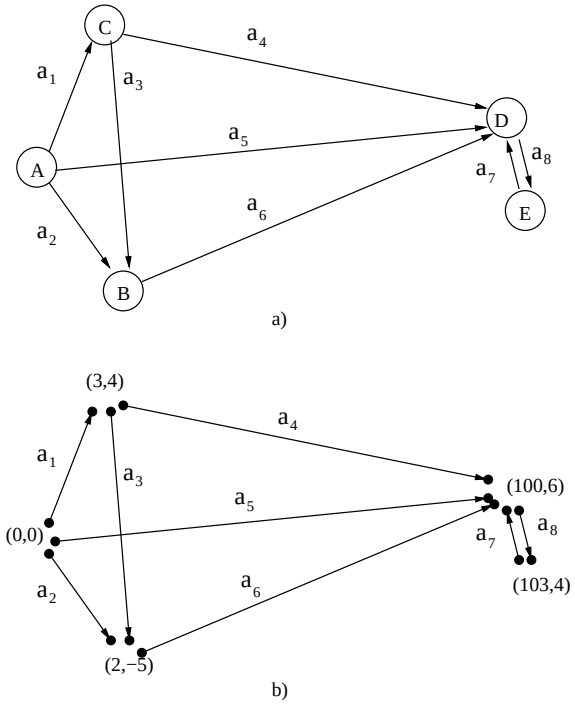

Figure 6: Simple WAN and its Constraint Graph

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
a_1		10.38	14.05	102.02	105.18	103.61	8.60	8.60
a_2			14.44	102.40	105.56	104.00	8.99	8.99
a_3				106.07	109.23	107.67	12.66	12.66
a_4					197.20	195.63	100.62	100.62
a_5						198.79	103.78	103.78
a_6							102.22	102.22
a_7								7.21
a_8								

Table 1: Constrained Distance Sum Matrix Γ, with $\Gamma(a_i, a_j) = d(a_i) + d(a_j)$.

obtained by merging the arcs a_4 with a_5 and a_6 in an optical link and implementing each of the other arcs with a dedicated radio link. The result is shown in Figure 7 where the dash-dot lines indicate a radio link and the solid line indicates an optical link.

If we change the application domain by moving to the problem of deriving an architecture for an on-chip communication network, the characteristics of the constraints and the cost function are quite different. Still the proposed approach can be used to find for instance the minimum number of repeaters (stateless buffers) that it is necessary to insert on a metal line while performing an optimum segmentation using the notion of critical length (l_{crit}) as defined in [12]. For this application, a first-cut library $\mathcal{L}$ can be considered as composed by only one link (a metal wire of lenght l_{crit} that is only dependent on the technology process) and three communication nodes (an inverter, a multiplexer and a de-multiplexer, all optimally sized). By using the Manhattan distance as the appropriate measure for the length of the links, the cost of each arc in the implementation graph is given by $\lfloor (|x_v - x_u| + |y_v - y_u|)/l_{crit} \rfloor$. Figure 8-(a) illustrates an example of this application, where we have studied the most critical channels on a multi-processor MPEG 4 decoder implemented in a 0.18μ technology. The final communication architecture, reported in figure 8-(b), has a total number of 55 required repeaters (with $l_{crit} = 0.6mm$). It is important to notice that this result is valid as long as the assumption that all links on the chip have a delay smaller than the clock period. Naturally, with the advent of deep sub-micron (DSM) process technology (0.13μ and below), this will be true for fewer wires. Still the approach presented in this work can be combined with the recently proposed latency-insensitive methodology [1], after making sure to define a cost function centered on the minimization of both stateless (buffers) and stateful (latches) repeaters.

5 Conclusions

This paper introduces a novel algorithm for the automatic synthesis of a communication architecture among a set of computational blocks once their relative positions and required pairwise communication bandwidth is provided. The algorithm is the result of a new way of modeling the problem (embodied by the notion of communication costraint graph) and it is based on a series of theoretical

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
a_1		9.05	14.05	102.02	97.02	102.40	200.09	200.17
a_2			5	103.61	98.61	104.00	201.69	201.58
a_3				98.61	103.61	107.67	198.61	198.42
a_4					5	9.05	100.00	100.63
a_5						5.38	103.07	103.78
a_6							101.40	102.22
a_7								7.21
a_8								

Table 2: Merging Distance Sum Matrix Δ, with $\Delta(a_i, a_j) = ||p(u) - p(u')|| + ||p(v) - p(v')||$.

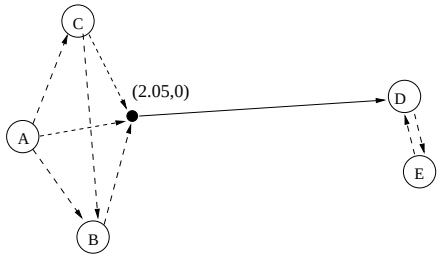

Figure 7: Example 1: Implementation Graph

results that give simple conditions to detect when a possible merging between point-to-point communication channels should be avoided because it is guaranteed to be part of a suboptimal solution.

Future work includes the extension to the case of statistical routing as well as to the case of shared use of resources. In the first case a channel use probability could be specified for each distinct point-to-point communications. The algorithm should be modified to consider their merging in case the available link bandwitdh is not fully exploited. The second case includes considering limitations in the number of library elements that can be deployed and/or their features (e.g. a switch that can not drive all its outgoing channels simultaneously).

Acknowledgments

This research was supported partially by the SRC and the GSRC/Marco center at Berkeley.

References

[1] L. P. Carloni, K. L. McMillan, A. Saldanha, and A. L. Sangiovanni-Vincentelli. A Methodology for *"Correct-by-Construction"* Latency Insensitive Design. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 309–315. IEEE, November 1999.

[2] C.P. Chang, P. Kermani, and A. Kershenbaum. Multi-link-speed network topology design. In *International Phoenix Conference on Computers and Communications*, pages 299–306, 1992.

[3] O. Coudert. On solving binate covering problems. In *Proc. of the Design Automation Conf.*, pages 197–202, June 1996.

[4] J.M. Daveau, T. B. Ismail, and A.A. Jerraya. Synthesis of system level communication by an allocation based approach. In *International Symposium on System Synthesis*, pages 150–155, 1995.

[5] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, 1979.

[6] Evguenii I. Goldberg, Luca P. Carloni, Tiziano Villa, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. Negative Thinking in Branch-and-Bound: the Case of Unate Covering. *IEEE Transactions on Computer-Aided Design*, 19(3):281–294, March 2000.

[7] K. Keutzer, S. Malik, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli. System level design: Orthogonolization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12), December 2000.

[8] Peter Voigt Knudsen and Jan Madsen. Integrating Communication Protocol Selection with Hardware/Software Codesign. *IEEE Transactions on Computer-Aided Design*, 18(8):1077–1095, August 1999.

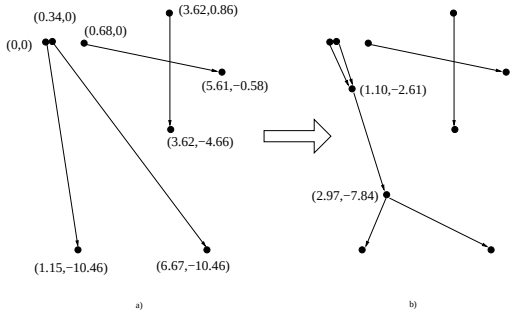

Figure 8: Example 2: Constraint graph and implementation graph

[9] K. Lahiri, A. Raghunathan, and S. Dey. Efficient exploration of the soc communication architecture design space. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 424–430, 2000.

[10] S. Liao and S. Devadas. Solving covering problems using LPR-based lower bounds. In *Proc. of the Design Automation Conf.*, June 1997.

[11] Li-Shiuan Peh and William J. Dally. Flit reservation flow control. In *International Symposium on High-Performance Computer Architecture*, pages 74–84, 1999.

[12] R. H. J. M. Otten and R. K. Brayton. Planning for Performance. In *Proc. of the Design Automation Conf.*, pages 122–127, June 1998.

[13] Ti-Yen Yen and Wayne Wolf. Communication synthesis for distributed embedded system. In *Proc. European Design Automation Conf.*, pages 288–294, November 1995.