Constraint-Driven Communication Synthesis

Alessandro Pinto[*]

Luca P. Carloni

Alberto L. Sangiovanni-Vincentelli

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.*

Categories and Subject Descriptors

J.6.1 [**Computer Applications**]: Computer-Aided Engineering—*CAD*.

General Terms

Algorithm.

Keywords

Communication Synthesis, Systems-on-Chip, Network Design.

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 unidirectional "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 authors are with UC Berkeley EECS Department, Berkeley, CA 94720-1772. E-mail: {apinto,lcarloni,alberto}@ic.eecs.berkeley.edu. This research was supported partially by the SRC and the GSRC/Marco Center at Berkeley.

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 bandwidth-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.

Among the several related papers published in recent years, the authors of [3] 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 [6] and [7] the design of the communication architecture is done with an exploration of different solutions validated by a fast performance simulation (based on a detailed characterization of the library components). The authors of [9] assume that the network topology is given and find an implementation that allows to achieve very high performance by 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 present one, but 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. Differently from our approach this paper assumes that the location of the intermediate communication nodes is fixed and the optimization is limited to link selection.
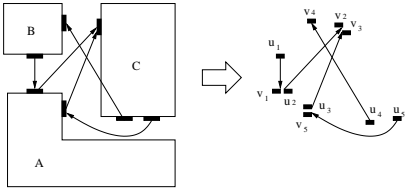
2. THE MODEL

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

Figure 1: Model of Communication Requirement.

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, 5] and we 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*: (1) the arc* length *(or,* distance*) $d(a)$ between vertex u and vertex v, and (2) the communication*

bandwidth $b(a)$ *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 vertices $a = |x_u - x_v| + |y_u - y_v|$. 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 node $n \in N$ has a cost $c(n)$. Each link $l \in L$ is characterized by a set of* link properties*: (1) the link* length *(or,* distance*) $d(l)$ corresponds to the length of the longest communication channel that can be realized by this link, (2) the link* bandwidth *$b(l)$ corresponds to the bandwidth of the fastest communication channel that can be realized by this link, and (3) the link* cost *$c(l)$, that is defined with respect to the other links in the library based on an optimality criterion that varies with the type of application.*

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. *Given a graph $G = G(V, A)$, a path $q = (v_1, a_1, v_2, a_2, \ldots, v_{Q-1}, a_Q, v_Q)$ 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: *the length $d(q) = \sum_{i=0}^{Q-1} d(a_i)$, the path bandwidth $b(q) = \min_{(i=0,\ldots,Q-1)} \{b(a_i)\}$, and the cost $c(q) = \sum_{i=0}^{Q-1} c(a_i)$, with $c(a_i)$ denoting the arc cost as 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 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. $\forall n' \in N', \exists n \in N(n = \psi(n'))$. The elements of N' are called* communication vertices.

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

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

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

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

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, 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.

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 is 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 is the concatenation of K library links interleaved by $k-1$ repeaters.*

- *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 is made by 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) 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 vertices). In fact, in general the cost of an implementation

---
[2] See also the assumption 2.1 defined below.

graph is smaller than the sum of the costs of its arc implementations, i.e., re-considering equation 1, we have:

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

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 $\mathcal{P}(a_1), \dots, \mathcal{P}(a_K)$ 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 $q^\star$, whose bandwidth $b(q^\star) \geq \max_{(k=1,\dots,K)}\{b(a_k)\}$ is called the common path of the merging transformation.*

After considering the possibility of K-way mergings, 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 $\mathcal{G}'(\mathcal{G}, \mathcal{L}) = (V' \cup N', A')$.*

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 $\mathcal{P}(a), \mathcal{P}(a')$:*

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

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

3. THE ALGORITHM

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 definition 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 "local minimum" that is encountered while searching the

```
GenerateCandidateArcImplementations (G, L)
{Compute set S of candidate arc implementations}
S ⇐ ∅
{Get optimum point-to-point arc implementations}
for all arc a ∈ G do
    P(a) ⇐ findBestPointToPointImplementation (a, L)
    S ⇐ S ∪ σ(P(a))
end for
{Find non-dominated candidate k-way mergings}
B ⇐ ComputeBandwidthVector (G)
Γ ⇐ ComputeConstrainedDistanceSumMatrix (G)
Δ ⇐ ComputeMergingDistanceSumMatrix (G)
k ⇐ 1
foundCandidateMapping ⇐ TRUE
while (foundCandidateMapping) do
    for all column j ∈ col(Γ) do
        kWayMergingNotFound ⇐ TRUE
        for all row subsets I = {i₁, ..., iₖ} ⊆ row(Γ) do
            if (∑ᵏᵢ₌₁ Δ[i, j] < ∑ᵏᵢ₌₁ Γ[i, j]) then
                b_min ⇐ min { B[j], min_{i∈[1,k]}{B[i]} }
                if (∃l ∈ L s.t.(b(l) + b_min) < ∑ᵏᵢ₌₁ B[i] + B[j]) then
                    S ⇐ S ∪ findKMergingImplementation(j, I)
                    kWayMergingNotFound ⇐ FALSE
                end if
            end if
        end for
        if (kWayMergingNotFound) then
            col(Γ) ⇐ col(Γ) \ {j}
        end if
    end for
    if col(Γ) = ∅ then
        foundCandidateMapping ⇐ FALSE
    else
        k ⇐ k + 1
    end if
end while
```

Figure 2: Deriving candidate arc implementations.

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

When combined, the two steps correspond to implicitly considering all possible communication sub-architectures that can be generated by putting together communication library components while satisfying the constraint graph requirements. To be effective, this approach must rely on the ability of generating the smallest 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 responsibility of finding the global optimum to state-of-the-art UCP solvers [4, 8], 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 present a set of theoretical results that guide us during the first step, thus 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 mergings, it is natural to focus on defining criteria that establish when a subset of K arcs should not 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 with 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 $\mathcal{G}^\star(\mathcal{G}^k, \mathcal{L})$ is a k-way merging. The set of sets of k-way mergeable arcs is denoted as $\mathcal{M}^k$.*

The following lemma provides a sufficient condition to detect when a pair of arcs is not 2-way mergeable [3]. 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. *Given a constraint graph $\mathcal{G} = G(V, A)$, let $A^2 = \{a, a'\} \subseteq A$, with $a = (u, v), a' = (u', v')$. If $d(a) + d(a') \leq ||p(u) - p(u')|| + ||p(v) - p(v')||$ then $A^2 \notin \mathcal{M}^2$*

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.

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 $\mathcal{G} = G(V, A)$. If*

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

then $A^k \notin \mathcal{M}^k$

The following theorem guarantees that if an arc a can not be part of any k-way merging then it also can not be part of any $(k+h)$-way mergings with $h \in [k+1, |A|]$.

THEOREM 3.1. *Let $\mathcal{G} = G(V, A)$ be a constraint graph. For all arc $a \in A$, if $\forall k \in [2, |A| - 2], \forall A^{k-1} \subset A \setminus \{a\}$, $(A^{k-1} \cup \{a\} \notin \mathcal{M}^k)$, then $\forall h \in [k, |A| - 1], \forall A^h \subset A \setminus \{a\}$, $(A^h \cup \{a\} \notin \mathcal{M}^{h+1})$.*

Given a constraint graph $\mathcal{G} = G(V, A)$ and a 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,*

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

Figure 2 illustrates the algorithm to generate a minimal set of candidate arc implementations that is based on the above results. 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, the algorithm proceeds by subsequently considering all possible k-way mergings for incrementing value of k. 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. The algorithm leaves the main

[3] The theorem proofs can be found in [10].

	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: $\Gamma(a_i, a_j) = d(a_i) + d(a_j)$.

	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: $\Delta(a_i, a_j) = ||p(u) - p(u')|| + ||p(v) - p(v')||$.

loop, when the set of columns of Γ becomes empty. The algorithm terminates returning the set $\mathcal{S}$ of candidate arc implementations, whose exact structures (i.e the exact topology, communication node position, number of links,...) are later obtained solving a simple nonlinear optimization problem, which computes also their costs. 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 1 if the implementation j implements the arc i, to 0 otherwise. Each columns has also a weight corresponding to the implementation cost. The selection of the optimum global solution corresponds to the solution of this instance of weighted Unate Covering Problem (UCP) and can be found by using state-of-the-art UCP solvers [4, 8].

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, Figure 3-(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 3-(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 2, 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
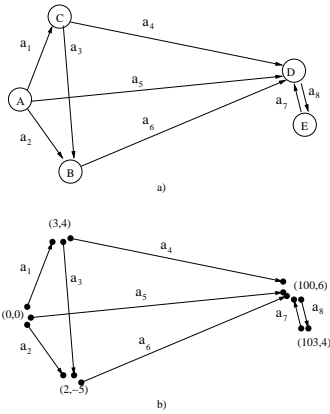
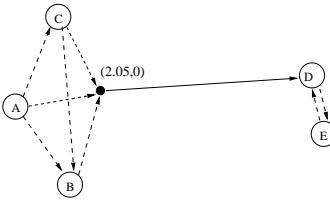**Figure 3: A Simple WAN and its Constraint Graph**

Figure 4: Example 1: Implementation Graph

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, twenty-one 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 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 4 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 [11]. For this application, a first-cut library $\mathcal{L}$ can be considered as composed by only one link (a metal wire of length 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 5-(a) illus-
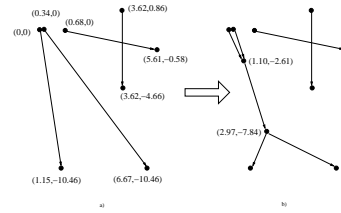

Figure 5: Final Implementation (Ex. 2).

trates 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 5-(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

We introduced 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 constraint graph) and it is based on a series of theoretical 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.

6. 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, Nov. 1999.

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

[4] E. I. Goldberg, L. P. Carloni, T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Negative Thinking in Branch-and-Bound: the Case of Unate Covering. *IEEE Transactions on Computer-Aided Design*, 19(3):281–294, Mar. 2000.

[5] 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.

[6] P. V. Knudsen and J. Madsen. Integrating Communication Protocol Selection with Hardware/Software Codesign. *IEEE Transactions on Computer-Aided Design*, 18(8):1077–1095, Aug. 1999.

[7] 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.

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

[9] L.-S. Peh and W. J. Dally. Flit reservation flow control. In *International Symposium on High-Performance Computer Architecture*, pages 74–84, 1999.

[10] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. Constraint-Driven Communication Synthesis. Technical Report available at www-cad.eecs.berkeley.edu/~lcarloni, Apr. 2002.

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