Synthesis of System-Level Communication by an Allocation-Based Approach

Jean-Marc Daveau, Tarek Ben Ismail, Ahmed Amine Jerraya,
TIMA/INPG Laboratory,
46, Av. Félix Viallet, 38031 Grenoble, France
E-Mail: daveau@verdon.imag.fr

Abstract

Communication synthesis aims to transform a system with processes that communicate via high level primitives through channels into interconnected processes that communicate via signals and share communication control. We present a new algorithm that performs binding/allocation of communication units. This algorithm makes use of a cost function to evaluate different allocation alternatives. The proposed communication synthesis approach deals with both protocol selection and interface synthesis and is based on binding/allocation of communication units. We illustrate through an example the usefulness of the algorithm for allocating automatically different protocols within the same system.

1. Introduction

As the synthesis community moves towards high-level synthesis, thus dealing with hardware/software codesign, some problems heretofore non existing appear [Wolf94]. One of these problems, which appears after system-level partitioning is the communication synthesis [BIJe95]. The goal of partitioning is to divide a system functionality into different subsystems where each subsystem is to be executed either in software or in hardware [GaVa95]. Communication synthesis is then needed because different subsystems inevitably need to communicate. This paper introduces a new approach for communication synthesis. This task is formulated as an allocation problem aimed at selecting a set of communication units from a library that implements the data exchange between the subsystems.

Previous work has focused on either protocol synthesis [SaPr90] or interface synthesis [NaGa94], but to our knowledge none of the existing work tackle communication synthesis as an allocation problem. The main contribution of this paper is to present communication synthesis as an allocation problem. An approach where communication is done through shared memory with a CSP type protocol has been presented in the literature [HeEr94]. In this approach, communication time is estimated through data flow analysis. The goal is to have a feedback that helps in meeting real time constraints. In our approach the detailed I/O structure and protocols are hidden in a library of communication components. The only visible part of these components are their services also called methods. This allows communicating processes to communicate by means of high level communication schemes. In other words theses processes use a remote procedure call mechanism to invoke communication services [Andr91].

In the following sections we present our proposed communication synthesis method. We will start by detailing the steps involved in communication synthesis. Then these steps will be described as an allocation problem. Next, we will describe our algorithm for channel unit allocation. Finally, we will present some results before concluding the paper.

2. Communication synthesis

Communication synthesis aims to transform a system with processes that communicate via high level primitives through channels into interconnected processes that communicate via signals and share communication control. As shown in figure 1, this activity includes two tasks:

- channel binding/allocation, and
- channel mapping, also called interface synthesis.

Figure 1a presents a conceptual communication over a communication network. This network is composed of a set of logical channel units. The channel binding/allocation task generates the system structure presented in figure 1b. The corresponding implementation resulting after the channel mapping task is detailed in figure 1c. The following part of the section details these steps.

A channel binding/allocation algorithm chooses the appropriate set of communication units from the library of communication in order to provide the desired services (s_i)

required by the communicating processes. The communication between the subsystems may be executed by one of the schemes (synchronous, asynchronous, serial, parallel) described in the library. The choice of a given communication unit will not only depend on the communication to be executed but also on the performances required and the implementation technology of the communicating processes. These features may be packed into a cost function to be reduced by the allocation algorithm. This is similar to the binding/allocation of functional units in classic high-level synthesis tools. Most of the allocation algorithms used in high-level synthesis may be used to solve this problem [MiLa92].

The channel mapping task replaces all communication units by distributing the communication protocol among the communicating subsystems and specific communication controllers. These controllers are selected from a library of channel implementations. A channel implementation is selected with regard to data transfer rates, memory buffering capacity, and the number of control and data lines. This task consists in mapping a logical communication structure onto a physical communication structure. The result of this task is a set of interconnected units through buses connecting the physical components and possible additional dedicated physical components (e.g. controllers, or bus arbiters). A unit may be an abstract processor or a component selected from the library of channels.

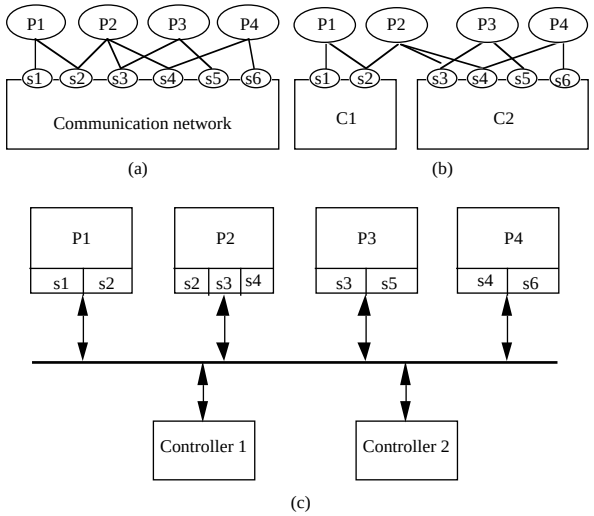

Figure 1: (a) Example of conceptual communication with processes cooperating through a communication network, (b) Corresponding system structure with processes cooperating via components (channels C1 and C2). (c) Corresponding physical implementation resulting after channel mapping.

3 Statement of communication as an allocation problem

In this paper, our goal is to combine both protocol synthesis and interface synthesis to have a complete communication synthesis method. In this approach, a system is specified as a set of communicating subsystems. We assume that the communication specification is separated from the rest of the design. At a high level of abstraction this communication is modelled as a *logical* communication network providing a set of services (also called methods or procedures), e.g. send/receive, used by the communicating processes (see figure 1a). In our model this logical network is composed of a set of channel units, each offering a certain number of services (s_i).

The communication synthesis is formulated as an allocation problem aimed at fixing the number and type of communication units needed to implement the logical network. Given (1) a set of processes communicating via a set of primitives and (2) a library of functional communication units with their member functions (methods), e.g. send, receive, and specific protocols, e.g. synchronous or asynchronous communication. The objective is to allocate (assign) a set of communication units, that perform the task of the logical network, to carry out the desired communication (see figure 1b). Each communication unit hides a special kind of communication implementation. This scheme allows more than one form of communication protocol to exist within the same framework. The communication synthesis allows to fix the implementation (actual physical links and possible communication components) of the communication scheme (see figure 1c). Currently there is no system in our knowledge that performs the selection of the physical communication structure automatically.

4 Communication modelling

In this paper we will use the communication modelling strategy described in [JeOB94]. A system is represented as a network composed of a set of abstract processors, called Design Unit (DU) communicating through a set of communication units known as Channel Units (CU). The channel unit allows communication between any number of design units. Channel units are based on the principle of remote procedure calls (RPC) [Andr91]. The networking services are transparent to the user and communication is invoked using the semantics of a standard procedure call.

In a conceptual view, the channel unit is an object that can execute one or several services (e.g. send, and receive) which may be blocking or non blocking. These services can share some common resources (e.g. a communication controller). For instance, the interaction

between these services and a controller modifies the state of the channel and synchronises the communication (see figure 2). This model enables the user to describe a wide range of communication schemes, ranging from a simple handshake protocol to a complex layered protocol, and most system-level communication such as message passing or shared resources.

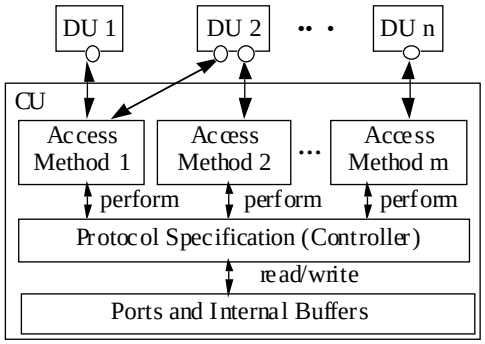

Figure 2: Solar's communication model: the channel unit.

Access to a channel unit is achieved by a fixed set of procedures, known as methods or services. These methods correspond to the visible part of the channel. In order to communicate, a design unit needs to access at least one method by mean of a special procedure call statement known as CUCall. The use of methods allows to hide the details of the channel unit. All accesses to the interface of the channel unit is made through these methods. Such methods also fix the protocol of exchanging parameters between the subsystem and the channel unit. Communication abstraction in this manner enables a modular specification, allowing communication to be treated independently from the rest of the design.

5. Channel allocation/binding algorithm

5.1 Introduction

The proposed allocation/binding algorithm starts with a logical network of channel units providing services to design units, and a library of functional communication units. The main task of the algorithm is to allocates a set of instances of communication units. Allocation is based on a cost function that is to be reduced and some constraints that have to be met. All methods belonging to the same channel unit represent an indivisible and coherent unit from the point of view of the protocol (they share the same controller), therefore they will be assigned on the same instance of the communication unit selected from the library (see figure 3).

For a given channel unit taken from the logical network, we will use the same set of constraints defined in [GaVa94] :

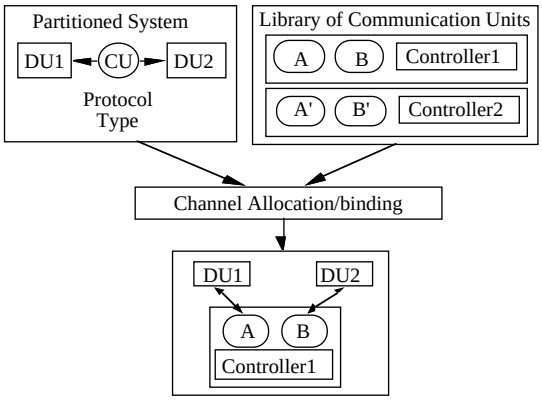

Figure 3: Channel allocation/binding.

- The protocol requested for the communication noted *Protocol.*
- The average transfer rate *AveRate* which is defined as the rate at which data flow over the bus. Any allocated communication unit must be able to provide this transfer rate.
- The peak transfer rate *PeakRate* which is defined as the rate at which a single transfer occurs over the bus.

Both *AveRate* and *PeakRate* are specified in bits/clock. Those constraints can be set by the user or given by an estimation tool.

With each communication unit from the library are given a set of properties :

- Its cost noted Cost which represents the intrinsic cost of the component due to its complexity, silicon area, buffering capacity, etc.
- The protocol implemented by that communication unit noted *Protocol.*
- The maximum bus rate *MaxBusRate* at which the data can be transferred across the communication unit.
- The maximum number of independent communications it can support noted *MaxCom*.

A communication unit can be a candidate for allocation if it satisfies the three following conditions:

- It provides the required services.
- It provides the right protocol.
- It provides the minimum required bus bandwidth.

In allocating communication units we attempt to assign several channel units on the same instance of a communication unit. If the channel units need to transfer data at a certain average rate, after being instanciated onto the same physical communication unit they should be able to transfer data at the same rate [NaGa94]. Therefore for a feasible solution we must have :

$$MaxBusRate \geq \sum_i AveRate_i$$

To ensure that a single data transfer doesn't take unnecessarily long time, the peak rate should be satisfied. This can be expressed as :

$$MaxBusRate \geq PeakRate_i, \forall\ i$$

As a finite number of channel units can be assigned on a

single instance of a communication unit we must have :

$$\sum_i(1) \leq \text{MaxCom}$$

For all logical channel units i allocated on the same instance of a communication unit.

5.2 Algorithm

The proposed algorithm first builds the tree of possible implementations. This decision tree enumerates for each logical channel unit all the physical communication units from the library that are candidate for allocation. The nodes of the tree are the logical channel units. Each node will have as many candidates as physical communication units that may implement that logical channel unit. The leaves of the tree correspond to empty nodes. Each path in the tree corresponds to a possible solution.

The second step of the algorithms perform a depth first exploration of the tree in order to select the best solution. Each path from the root to a leaf is a possible solution. In order to handle the case where several logical channel units are assigned on the same instance of a library communication unit, we use a procedure called *merge*. This procedure is used during the tree exploration in order to assign several logical channel units on the same instance of library a communication unit. If it fails a new instance of a communication unit will be created.

The algorithm is detailed in the rest of this section. The main program builds the tree. The procedure *traversal* explores the tree and procedure *merge* allows the allocation of several channel units on the same component during the tree exploration.

The cost function to be reduced by the allocation algorithm takes into account a selected communication unit B_j from the library and several channel units M_i to be instanciated on it. This cost function is given below:

$$\text{cost function} = K_1 * \text{cost}(B_j) + K_2 * \sum_i [\text{PeakRate}(M_i) - \text{MaxBusRate}(B_j)]^2$$

The second term of the cost function is taken into account only if the constraint on PeakRate is violated, i.e. only if $\text{PeakRate}(M_i) > \text{MaxBusRate}(B_j)$.
K_1 and K_2 are user set parameters used to weighs each term of the cost function. These allow trade-offs between component cost and performance.

Let M_i be a logical channel unit offering a set of methods $\text{Methods}(M_i)$ that have to be allocated on the same physical communication unit. Let B_j be an element of the library of communication units. B_j is a physical communication unit that offers a set of methods $\text{Methods}(B_j)$. Let $\mathfrak{R}$ be a solution for the allocation/ binding of the logical communication network and total_cost its cost. Let $\mathfrak{I}$ be a list of instances of communication units that have already been allocated along a path in the tree: $\mathfrak{I}=\{I_1, I_2, .. I_\varphi\}$. With each I_k come a set of variables:

- The current bus load of that communication unit noted $\text{BusRate}(I_k)$. It is the sum of the AveRate of all channel units allocated on that instance.
- The number of communications handled called $\text{CurrentCom}(I_k)$.

With each node of the tree is associated a logical channel unit noted LogicalChannel(*node*) and to each outgoing decision edge a physical communication unit noted CommunicationUnit(*edge*).

ALGORITHM

```
Procedure MERGE(InstanceList ℑ, Instance CU,
                Instance V, Integer merge_cost) {
merge_cost = + ∞
For each instance Ik == CU, Ik∈ℑ Do {
   If protocol(CU) = protocol(Ik) and
   BusRate(Ik)+AveRate(Mi) ≤ MaxBusRate(Ik) and
   CurrentCom(Ik)+1 ≤ MaxCom(Ik) Then {
      If PeakRate(Mi) > MaxBusRate(Ik) Then {
         current_merge_cost =
         K2*[PeakRate(Mi)-MaxBusRate(Ik)]^2
      Else
         current_merge_cost = + ∞ }
      If current_merge_cost < merge_cost Then {
         merge_most = current_merge_cost
         V = Ik } } }
Return V and merge_cost }

Procedure TRAVERSAL( Node n, InstanceList ℑ,
                     Integer current_cost) {
If n is a leaf Then {
   If current_cost < total_cost Then {
      ℜ=ℑ
      total_cost = current_cost }
Else
   V = Ø
   Mi=LogicalChannel(n)
   For every edge e of n Do {
      CU = CommunicationUnit(e)
      MERGE(ℑ, CU, V, merge_cost)
      If V ≠ Ø Then { /* merge successful */
         bind Mi to V
         BusRate(V) += AveRate(Mi)
         CurrentCom(V)++
         current_cost += merge_cost
         TRAVERSAL(e.nextnode, ℑ, current_cost)
      Else /* create a new instance */
         If AveRate(Mi) ≤ MaxBusRate(CU) Then {
            If PeakRate(Mi) > MaxBusRate(CU) Then {
               alloc_cost = K1*Cost(CU) +
               K2*[PeakRate(Mi)-MaxBusRate(CU)]^2
            Else /* no constraint violated */
               alloc_cost = K1*Cost(CU) }
```

```
        bind M_i to CU
        current_cost +=alloc_cost
        BusRate(CU)=AveRate(M_i)
        CurrentCom(CU)++
        J+={CU}
        TRAVERSAL(e.nextnode, J, current_cost)
    Else /* not a feasible solution */
        current_cost = + ∞ }}}}}
```

Algorithm Allocation/Binding **{**
build the decision tree
A = {Ø} total_cost = + ∞ current_cost = 0
TRAVERSAL(/, J, current_cost) **}**

6. Results

In this section we show the results of applying our allocation algorithm onto a simple example. We consider a *send-and-receive* system. It is composed of two subsystems, a server, a host and two channel units ensuring a bi-directional communication. Let's assume that the communication synthesis starts with the system of figure 4. We give the following set of constraints on the channel units:

- AveRate(CU_host) = 12 bits/clock, and PeakRate(CU_host) = 18 bits/clock.
- AveRate(CU_server) = 4 bits/clock, and PeakRate(CU_server) = 8 bits/clock.
- Protocol(CU_host) = Protocol(CU_server) = *any*.

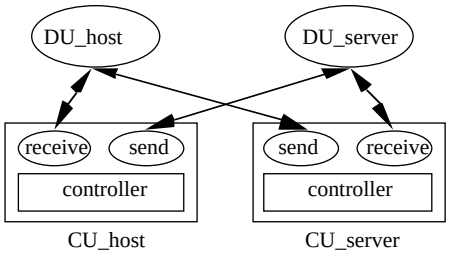

Figure 4: *Send-and-receive* system.

We set K_1 = 1 and K_2 = 10, therefore we privilege the performance by setting its weight to 10 times the weight of the component. We assume also that we start with a library including three communication units:

- a CU with a double RDV protocol.
- a CU with a queue.
- a CU with a double queue.

The characteristics of each communication unit are outlined in figure 5. Figure 6 shows the decision tree corresponding to the system described in figure 4 with the communication units library of figure 5. Five of the possible allocation/binding alternatives are listed below and described in figure 7. The leaf nodes indicate the cost before/after merge.

a- both CU_Host and CU_Server with a single bus, and total_cost = 60.
b- both CU_Host and CU_Server with a queue, and total_cost = 240.
c- CU_Host with a bus and CU_Server with a queue, and total_cost = 160.
d- CU_Host with a queue and CU_Server with a bus, and total_cost = 200.
e- both CU_Host and CU_Server with a double queue, and total_cost = 170.

These alternatives make use of the functional communication units library presented in figure 5. The total_cost is obtained by applying the cost function detailed above. The algorithm is going to choose the solution with the lowest cost, therefore solution (a) will be retained. The two logical channel units will be physically implemented as a single bidirectionnal bus that multiplexes both accesses from DU_host and DU_server. The channel mapping is going to map a predefined generic interface onto the design units. Thus it will generate one bus with its control signals (see figure 8). This step generates all the interfaces. This corresponds to an expansion of procedure calls into the design units according to the communication units selected. The size of the bus will be fixed by the channel mapping algorithm depending on the data transfer rate. An approach for determining the width of a bus that will implement a group of channels is presented in [FiKu93].

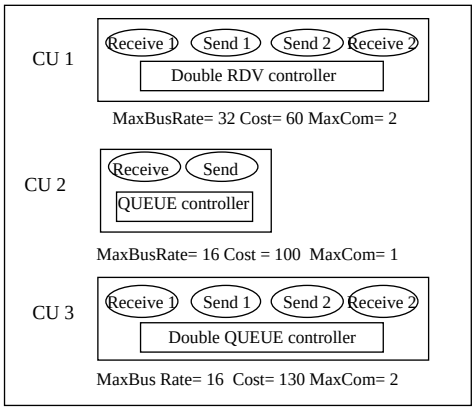

Figure 5: Functional communication units library.

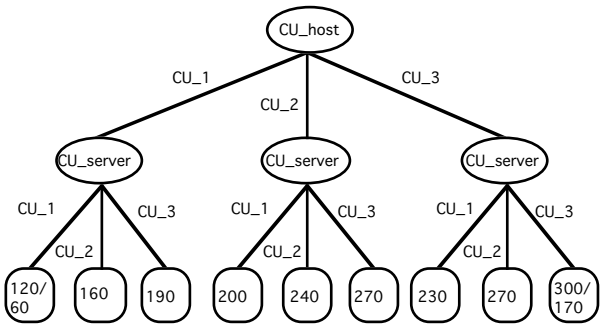

Figure 6: Decision tree for the allocation/binding of system *send-and-receive.*

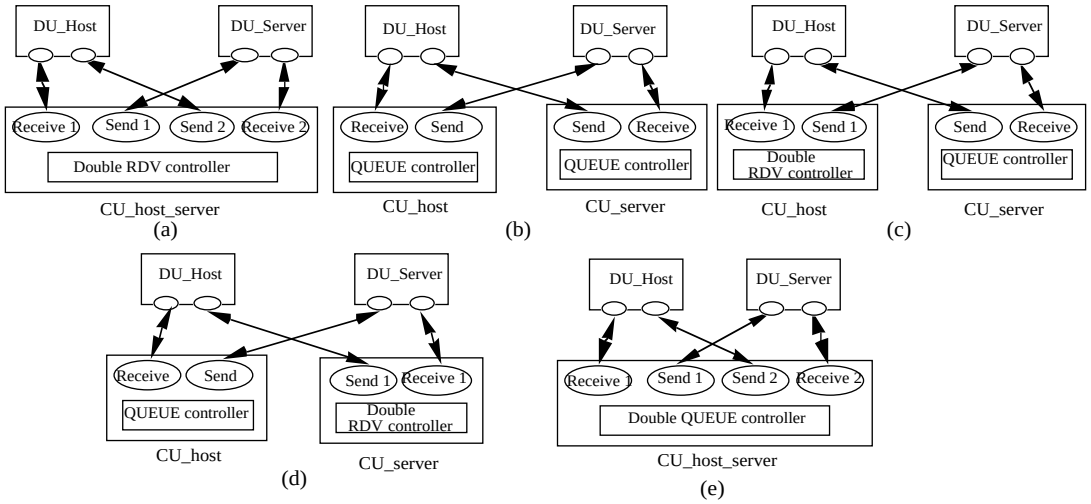

Figure 7: Channel allocation/binding alternatives.

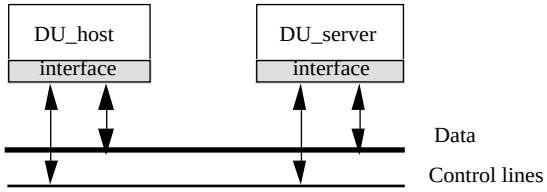

Figure 8: System *send-and-receive* after channel mapping.

7. Conclusions

We have presented in this paper a means whereby communication synthesis is stated as an allocation problem. This problem can be solved by using most of the allocation algorithms used in high level synthesis. We have presented one possible algorithm for channel unit allocation/binding. This algorithm is based on a decision tree. The channel mapping task is performed automatically. The key issue in this scheme is the use of an abstract and general communication model. The separation between communication and computation allows the reuse of existing communication models. A library of channels offers a wide range of communication mechanisms allowing the designer to select the appropriate communication protocol for his application. Since no restrictions are imposed to the communication models, this codesign process can be applied to a large class of applications.

References

[Andr91] G.R. Andrews, "Concurrent Programming, Principles and Practice", Benjamin/ Cummings (eds), Redwood City, Calif., pp. 484-494, 1991.

[BIJe95] T. Ben Ismail, and A.A Jerraya, "Synthesis Steps and Design Models for CoDesign", IEEE Computer, special issue on rapid-prototyping of microelectronic systems, Vol. 28, No. 2, pp. 44-52, February 1995.

[FiKu93] D. Filo, D. Ku, C. Coelho, and G. de Micheli, "Interface Optimisation for Concurrent Systems Under Timing Constraints", IEEE Trans. on VLSI, Vol. 1, pp 268-281, September 1993.

[GaVa95] D. Gajski, and F. Vahid, "Specification and Design of Embedded hardware/software Systems", IEEE Design & Test of Computers, pp. 53-67, Spring 1995.

[HeEr94] J. Henkel, R. Ernst, U. Holtman,and T. Benner, "Adaptation of Partitioning and High Level synthesis in hardware/Software Co-Synthesis", Proc. ICCAD, pp. 96-100, November 1994.

[JeOB94] A.A. Jerraya, and K. O'Brien, "Solar: An Intermediate Format for System-Level Modeling and Synthesis", in "Computer Aided Software/Hardware Engineering", J.Rozenblit, K.Buchenrieder (eds), IEEE Press, Piscataway, N.J., pp 147-175, 1994.

[MiLa92] P. Michel, U. Lauther, and P. Duzy, "The Synthesis Approach to Digital System Design", Kluwer Academic Publishers, 1992.

[NaGa94] S. Narayan, and D. Gajski, "Synthesis of System-Level Bus Interfaces", Proc. EDAC'94, Paris, France, pp. 395-399, February 1994.

[SaPr90] K. Salah, and R. Probert, "A Service-Based Method for the Synthesis of Communication Protocols", Int'l Journal of Mini and Microcomputers, Vol. 12, No. 3, pp. 97-103, 1990.

[Wolf94] W. Wolf, "Hardware/Software Co-Design of Embedded Systems", Proc. IEEE, Vol 82, No 7, pp 967-989, 1994.