8

Interface Synthesis in Embedded HW/SW Systems

G. Gogniat, M. Auguin, C. Belleudy
I3S, Université de Nice Sophia/Antipolis - CNRS, 41 Bld. Napoléon III 06041 Nice Cedex, France.
Phone: 33 - 04 93 21 79 58 Fax: 33 - 04 93 21 20 54
E-mail: gogniat@alto.unice.fr

Abstract

Since several years, complex embedded systems are used in an expanding number of domains. Complexity and time to market constraints impose to introduce major improvements in CAD methodologies. Hardware-Software codesign attempts to deduce automatically heterogeneous system solutions from a high level specification. The interface between heterogeneous resources can become critical in time and/or area for telecommunication applications, hence it is important to define techniques that minimize the communication overhead cost. In this paper we present an original method to synthesize efficient interfaces.

Keywords

communication synthesis, template architecture, codesign, telecommunication

1 INTRODUCTION

The complexity of embedded systems for multimedia and wireless applications imposes the use of heterogeneous resources (i.e. processor cores, dedicated functional units). Codesign methodologies of such systems must respect cost and performance constraints with a reduced time to market. The need for codesign techniques results from the increasing complexity of applications and the advances in HW and SW technologies. Due to the lack of a general formalism able to provide models for various application domains with efficient HW or SW implementations, codesign methodologies concentrate on a specific application domain with a dedicated generic architecture. Since we focus on telecommunication and multimedia applications we consider a data flow oriented static model that permits to describe a wide range of applications in this area. Codesign methodologies target a template or a generic architecture composed of interconnected heterogeneous resources. For embedded system design, it is of prime importance to develop methods for minimizing the area and delay introduced by the interconnection. The following section depicts characteristics of our template architecture and the communication model. Section 3 presents a communication synthesis method that promotes synchronous transfers in that archi-

tecture to reduce the interconnect area. Before concluding, results about an acoustic echo canceller are depicted.

2 TEMPLATE ARCHITECTURE AND COMMUNICATION MODEL

Our target architecture is based on the DSPA (Data Synchronized Pipeline Architecture) architecture. This architecture is composed of asynchronous functional units (FUs). A functional unit is either a specific cell, a processor core or a synthesized cell. In this architecture, interconnection crosspoints between units are modeled by FIFO queues (Auguin, 1996). This asynchronous communication model leads to an ASAP (As Soon As Possible) execution style of operations by FUs. Since considered applications have a static behavior, a fixed scheduling of operations may be defined during partitioning. Then, some communications may be changed into synchronous transfers (*rendez-vous* mechanism) without overstepping timing constraints. The protocol associated with a transfer can be blocking or non blocking. The protocol is blocking if before reading or writing into a crosspoint the FU must check availabilities of data. With a non blocking transfer no verification needs to be done. To limit the cost due to FIFOs, the static schedule of tasks has to minimize the use of asynchronous communications. This point is addressed in the following section.

3 COMMUNICATION SYNTHESIS

A static data flow model of an application can be described by a DAG (Directed Acyclic Graph) where nodes correspond to tasks of the application and edges represent data transfer between tasks. After the partitioning step communication edges represent links between HW and SW units, between different SW units or different HW units. Communication synthesis consists in determining for each communication edge the type of transfer (i.e. synchronous or asynchronous), the communication resources, the transfer protocol (blocking or non blocking) and the transfer mode (DMA or memory mapped I/O for processors). As mentioned above, the template architecture considers initially asynchronous communications with FIFOs. Hence, it is of prime importance to minimize their use in order to reduce communication resources. In the sequel, we address this point. The communication synthesis method assumes that after partitioning a schedule of tasks on FUs is provided. The aim is to transform asynchronous communications into synchronous ones by local reschedulings.

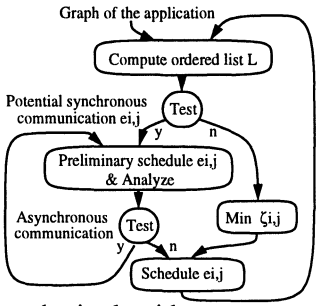

Figure 1 Communication synthesis algorithm

The algorithm (Figure 1) operates as follows. Firstly, nodes and edges are characterized. An edge $e_{i,j}$ is labelled when a transfer type (synchronous or asynchronous)

is assigned. Edges with asynchronous communications are labelled and are not considered for the remainder. An ordered list L of potential synchronous edges is created according to a cost function that define the priority of each communication edge. Nodes V_i and V_j corresponding to the first non labelled edge $e_{i,j}$ of L are preliminarily scheduled (local rescheduling). Impacts of this schedule on other communication edges is analyzed by characterizing nodes and edges again. If any communication edge $e_{k,l}$ ($k \neq i$ and $l \neq j$) becomes asynchronous, $e_{i,j}$ is definitively scheduled and is labelled with a synchronous transfer. Otherwise another non labelled edge $e_{i,j}$ from L is considered. The process is iterated until all the communication edges that have no impact on other edges are labelled. After this step remaining potential synchronous edges in L involve at least one asynchronous communication. The edge $e_{i,j}$ of L that conducts to minimize hardware communication resourcesis labelled with a synchronous transfer. This process is iterated until all nodes are labelled. After this step all the communications of the graph are characterized. Hence, communication resources, transfer protocols and transfer modes associated with each data transfer can be determined in order to complete the communication synthesis flow.

4 EXAMPLE: GMDFα

To illustrate the principles of this generic architecture we consider a frequency domain block adaptative algorithm for acoustic echo cancellation (GMDFα) (Freund, 1996). Before synthesis of communications all units are connected through FIFOs placed in a network of six bus. The schedule of application nodes allows to label all communication edges with a synchronous transfer mode avoiding FIFOs. With the knowledge of timings of data transfers, a minimization of the number of bus can be performed and the network can be reduced to only two bus. The final implementation meet timing and area constraints of the specification.

5 CONCLUSION

Communication synthesis represents one of the main step in the hardware-software system synthesis flow. Its aim is to define an interface supporting all the communications between heterogeneous resources. The communication interface must be adapted to the target architecture in order to obtain an efficient hardware-software implementation. The considered communication model is based on bufferized (through FIFOs) and non bufferized (bus) resources that allows to adjust the communication resources according to the transfer requirements. The method is based on performing a global optimization of all communications in order to minimize hardware area and to respect timing constraints.

6 REFERENCES

Auguin M., Belleudy C., Gogniat G., Jegou Y. (1996) A multi-granularity data synchronized architecture for HW/SW embedded DSP systems. *Proceedings Int. Conference on Signal Processing Applications & Technology*. Boston, october 7-10.

Freund L., Israel M., Rousseau F., Berge J.M., Auguin M., Belleudy C., Gogniat G. (1996) A codesign experiment in acoustic echo cancellation: GMDFα. *Int. Symposium on System Synthesis*. IEEE-ACM, La Jolla California, November 6-8.