Communication synthesis and HW/SW integration for Embedded System Design

Guy Gogniat, Michel Auguin, Luc Bianco, Alain Pegatoquet
I3S - Université de Nice Sophia Antipolis - CNRS
41 Bd Napoléon III - 06041 Nice cedex
email: *lastname*@alto.unice.fr

Abstract

The implementation of codesign applications generally requires the use of heterogeneous resources (e.g., processor cores, hardware accelerators) in one system. Interfacing hardware and software components together and providing communications between them are particularly error proned and time consuming tasks. Hence, on the basis of a generic architecture we propose an extended communication synthesis method that provides characterization of communications and their implementation scheme in the target architecture. This method takes place after partitioning and scheduling and can constitute the basis of a back end of a codesign framework leading to HW/SW integration.

1. Introduction

The evaluation of several hardware/software mixed solutions is generally essential to obtain an efficient implementation of a codesign application. These evaluations are based on different underlying architectures and heterogeneous components. Thus, designers need to define specific communication interfaces and the associated mechanisms for each solution. This step of the codesign flow is particularly time consuming and error proned. Therefore, it is essential to propose new methodologies dealing with communication synthesis to promote fast system prototyping. The aim is to characterize the communications of the application and to minimize the corresponding resources. Furthermore, the determination of the protocols, which may become difficult for complex systems using heterogeneous resources, is solved by the synthesis algorithm. Hence, starting from the specification, the communication synthesis step provides all the information needful to reach the HW/SW integration phase. Several studies handling this problem have already been published. In Yen and al. [1995] the developed heuristic leads to the determination of the communication resources (e.g., size and number of buses) and their corresponding schedule on the target architecture. However, they do not provide interface structures and control mechanisms to manage the communications. In Freund et al. [1997a] the evaluation and the scheduling of the communications is also done. Furthermore the protocols (i.e., synchronous or asynchronous) are determined. Works proposed by Gong et al. [1996] are based on four target architectures with local and global memories. Their method uses generic protocols and conducts to the final implementation of the application on the target architectures. The most relevant work to our problem is presented in Filo et al. [1993]. In order to minimize control logic on channels their interface optimization attempts to maximize the use of the non-blocking protocol. But the side effects consist in the introduction of control delays in both sender and receiver that impose a global reference clock. Moreover, most of the studies only consider a target architecture composed of a processor and an ASIC which may not be sufficient for complex embedded systems.

2. Underlying architectures

In our approach, we focus on the last steps of a codesign flow dedicated for static digital signal processing applications. These applications generally require the use of several processor cores (e.g., DSP, RISC or microcontroller), coprocessors and hardware accelerators. Hence, the underlying architecture of our communication synthesis method is composed of heterogeneous components [Gogniat 1997]. Computations between these components can be performed in parallel for higher performance. Furthermore, a component can execute its computations and communications either in a sequential or in a parallel mode through the use of a dual internal memory. Since, components of the architecture may result from previous designs we consider that each unit is encapsulated and communicates with well defined protocols (i.e., synchronous and asynchronous).

3. Communication synthesis

Based on the considered architecture, the last codesign phases include the communication synthesis and the HW/SW integration steps. In this approach the communication synthesis problem is addressed as a characterization and an optimization of communications involved in the application. Compared to Freund et al. [1997b] the extended synthesis method detailed in this paper is based on a refined communication model achieving accurate synthesis results. Communication synthesis and HW/SW integra-

tion take place after the partitioning and scheduling phase allowing the use of various partitioning heuristics.

3.1. Graph model of partitioned applications

Static signal processing applications can be modeled by a direct acyclic graph, where nodes represent computations and edges correspond to data dependencies. We note $G(V,E)$ the graph of the application after partitioning and scheduling. An edge $e_{i,j} \in E$ between two nodes V_i, $V_j \in V$ denotes a dependency. Three types of dependency are considered: temporal, functional and internal. A temporal dependency (TD) edge connects two nodes which do not communicate but are allocated to the same component. This link describes the order of execution of nodes imposed by the scheduling. Functional dependency edges (FD) represent data transfers between nodes. These edges are annotated with the volume (V_{data}) and the word size (L_{data}) of transferred data. An internal dependency (ID) edge connects two communicating nodes implemented on the same component. In this case, no transfer is needed since the data are stored in the internal memory of the component.

3.2. Communication synthesis flow

The aim of the communication synthesis steps (Figure 1) is to minimize the number of communication resources and the overhead delays due to communications while respecting specification constraints. The communication model used for the synthesis is based on the target architecture and is characterized by the following features:

- The transfer mode (i.e., parallel or sequential)
- The transfer type (i.e., synchronous or asynchronous)
- The communication supports and protocols (i.e., FIFO, bus - blocking, non blocking)

With the parallel transfer mode a communication and a computation thread can be executed in parallel using the internal data memories associated with each component of the architecture. The transfer type corresponds to the execution schemes of communications (i.e., synchronous or asynchronous). The communication supports represent the hardware resources required to implement the data transfers (i.e., FIFO, bus). The protocol associated with a communication can be blocking or non blocking. A blocking protocol must be selected if the data consistency is not guaranteed with a non blocking protocol. Figure 1 represents the synthesis flow which is divided in two main tasks. The first one performs the characterization of each communication of the partitioned and scheduled graph.

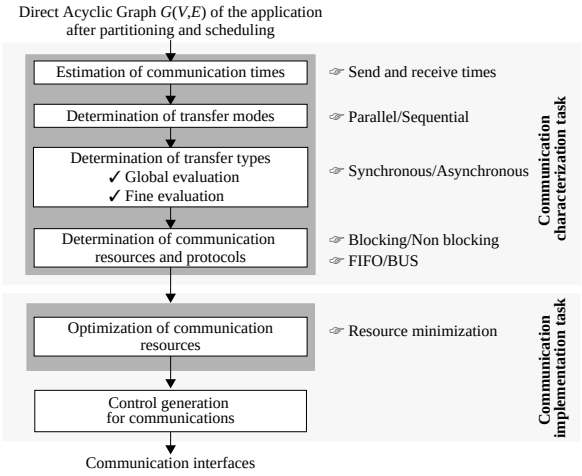

Fig. 1 • Communication synthesis flow

The second task optimizes the number of resources and determines the set of controls required to manage the communications. This last task leads to the complete overall definition of the final architecture.

3.2.1. Characterization of communication edges

The main contributing step in the characterization of communications is the transfer type determination since this step maximizes the number of synchronous communications and consequently minimizes the number of FIFO. Two methods of evaluation have been addressed in order to reach this goal. They consider respectively a global and a fine behavioral model of the system. The global one introduced in Freund et al. [1997b] is based on a usual communication and execution model [Lee and Messerschmitt 1987]. As shown in Figure 2(b) this model merges in a single time frame the computation (i.e., $Texe_i$) and communication times (i.e., send times $te_{i,j}$ and/or receive times $tr_{i,j}$) associated with a node of the graph. For example the node V_1 sending data to V_3,V_4,V_5 has a new execution time corresponding to its computation time increased by the three data emission delays. With this model, the execution of a node can only starts when all the data produced by its preceding nodes are transferred. Hence, the communication synthesis is based on a global model of communication. In the fine evaluation method, the communication model does not merge computations and communications leading to a more realistic behavioral description of the system (Figure 2(c)). Furthermore, a single communication time $tsyn_{i,j}$ is considered corresponding to the elapse time induced by the data transfer. This fine model permits a global and a local optimization of the communications since each data transfer associated with a node is locally ordered before the scheduling of the whole sequence of transfers. This point is detailed in the

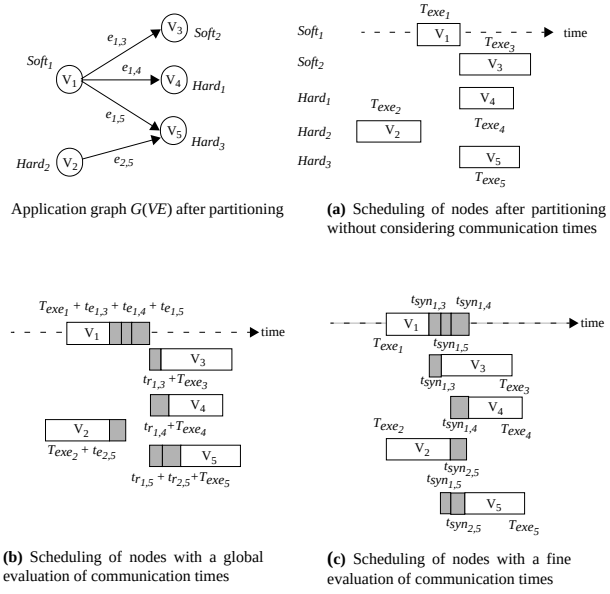

Fig. 2 • Execution schemes of communications

sequel. Since the global evaluation approach has been published in Freund et al. [1997b], we focus mainly on the fine evaluation method.

The communication synthesis starts with an estimation of all the transfer times associated with the communication edges (i.e., functional dependencies) of the partitioned graph. These communication edges correspond to data transfers between the different instances of the architecture (e.g., DSP and RISC, DSP and co-processor). With the considered target architecture these estimations represent the data transfer durations imposed by the internal memories of the sender ($te_{i,j}$) and the receiver ($tr_{i,j}$) and are given by:

$$T_{com} = V_{data} \times \left\lceil \frac{L_{data}}{L_{bus}} \right\rceil \times N_c \times T_c$$

where T_{com} means either $te_{i,j}$ or $tr_{i,j}$, V_{data} represents the number of data to be transferred between the sender and the receiver, L_{data} corresponds to the size of data, N_c is the number of clock cycles to access a data in the internal memory, T_c is the clock period of the communication interface and L_{bus} corresponds to the internal memory bus size. As the aim of the communication synthesis is to promote synchronous communications, the communication time $tsyn_{i,j}$ between two nodes V_i and V_j is estimated as the maximum of the sending and the receiving times i.e., the slowest interface imposes its communication rate with a handshake protocol.

The fine evaluation model illustrated in Figure 2(c) leads to reschedule nodes of the graph in order to take into account data transfers between components. This is achieved by computing a preliminary schedule of data transfers involved in a sequence of communications associated with a node. The order of analysis of these data transfers is given by scheduling rules. These rules schedule first the outputs of data corresponding to functional dependency edges (Figure 3(a)). We assume that the execution time of a node depends on the amount of consumed data. Therefore, the highest priority level $P_e(i,j)$=1 is associated with the edge with the longest communication time in order to promote executions of time consuming nodes. Priorities of other edges are set up according to their decreasing communication times. On the contrary, the receipt of data can only starts if temporal and internal dependencies are verified, hence the highest priority level $P_r(i,j)$=1 is associated to these edges (Figure 3(b)). For remaining edges a decreasing rule is also used.

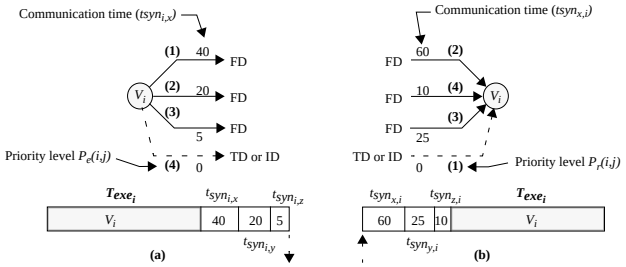

Fig. 3 • Priority levels of the senders and receivers

The local scheduling of communications attempts to maximize the use of synchronous transfers. However, even with local rescheduling it is not always possible to use only synchronous communications [Gogniat 1997]. In such cases, asynchronous communications are required and both transfer types are implemented in the final architecture. The algorithm is based on two functions: Node_characterization and Edge_characterization. For each node V_i, Node_characterization computes its mobility interval Δ_{M_Vi} defined as the interval between the ASAP starting time ts_{ASAP_i} and the ALAP ending time te_{ALAP_i} of V_i. Computations of interval mobilities take into account timing constraints and local scheduling of communications associated with nodes. The determination of ASAP starting time and ALAP ending time of nodes is presented in Figure 4. For each node V_i, all the predecessors and all the successors are scanned in order to define respectively ts_{ASAP_i} and te_{ALAP_i}.

Edge_characterization computes the mobility value $Me_{i,j}$ of a communication edge $e_{i,j}$ defined as the difference between te_{ALAP_i} and ts_{ASAP_i}. If $ts_{ASAP_i} > te_{ALAP_i}$ then

$Me_{i,j}$ is negative and the communication is asynchronous since there is no timing overlap between the sender and the receiver. Otherwise the communication is considered as potential synchronous. The Edge_characterization function also provides a cost value $\xi e_{i,j}$ for the edges with a potential synchronous communication which represents the ratio of the amount of data (V_{data}) transferred through this edge and its mobility value. Edges with the highest cost values are considered first since a better hardware minimization is expected if a synchronous transfer model is associated with these edges.

For all the nodes $V_x \in$ pred(V_i) $tval = te_{ASAP_x} + \sum tsyn_{x,j}$ ($V_j \in$ suc(V_x), $j \neq i$ and $P_e(x,j) > P_e(x,i)$)

  If $tval > ts_{ASAP_i}$ **Then** $ts_{ASAP_i} = tval + tsyn_{x,i}$;
  End if;

  If $tval < ts_{ASAP_i}$ **Then** $ts_{ASAP_i} = ts_{ASAP_i} + tsyn_{x,i}$;
  End if;

End for;

For all the nodes $V_x \in$ suc(V_i) $tval = ts_{ALAP_x} - \sum tsyn_{x,j}$ ($V_j \in$ pred(V_x), $j \neq i$ and $P_e(x,j) < P_e(x,i)$)

  If $tval < te_{ALAP_i}$ **Then** $te_{ALAP_i} = tval - tsyn_{x,i}$;
  End if;

  If $tval > te_{ALAP_i}$ **Then** $te_{ALAP_i} = te_{ALAP_i} - tsyn_{x,i}$;
  End if;

End for;

Fig. 4 • Determination of ASAP starting time and ALAP ending time of nodes

The algorithm for the transfer type determination with the fine evaluation method is similar to the one with the global evaluation model [Gogniat 1997] (except the evaluation of Node_characterization and Edge_characterization functions) and is briefly described (Figure 5). Firstly, nodes and edges are characterized. An edge $e_{i,j}$ is labelled when a transfer type (synchronous or asynchronous) is assigned to this edge. Edges with asynchronous communications (i.e., $Me_{i,j} < 0$) are labelled and are not considered for the remainder. Nodes V_i and V_j corresponding to the first non labelled edge $e_{i,j}$ are preliminary scheduled (local rescheduling). Impacts of this schedule on other communication edges are 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. Let $\zeta_{i,j}$ be the cost function associated with $e_{i,j}$ defined as the ratio of the total volume of data associated with edges of L that become asynchronous and the total volume of data associated with edges of L that remain synchronous. The edge $e_{i,j}$ of L with $\zeta_{i,j}$ minimum is labelled with a synchronous transfer since the objective is to minimize the area dedicated to FIFOs.

While all the communication edges are not labelled **do**
  For each potential synchronous communication edge in the list L **do**
    Preliminarily schedule the next edge $e_{i,j}$ that is not labelled;
    Analyze the impact of that solution on other communication edges;
      If no asynchronous communication edges is revealed **then**
      Schedule definitively the edge $e_{i,j}$;
      Label the edge $e_{i,j}$ with a synchronous transfer;
      Reorder the list L;
      End if;
  End for;
  Definitively schedule the edge $e_{i,j}$ that have the lowest cost function ζi,j;
  Label the edge $e_{i,j}$ with a synchronous transfer;
  Remove asynchronous communication edges from the list L;
  Reorder the list L;
End while;

Fig. 5 • Algorithm for transfer type determination

The next step in the communication synthesis flow outlines the nature of communication protocols and resources. For a synchronous communication only one bus is required to support the data transfer between the sender and the receiver. With an asynchronous communication three resources are necessary: the bus from the sender to the FIFO, the FIFO itself and the bus from the FIFO to the receiver. All these resources are characterized with their width, throughput and the depths for the FIFOs (Table 1). A sender and a receiver involved in a synchronous transfer use a blocking protocol. With an asynchronous transfer the receiver can use either a blocking or a non-blocking protocol (see Section 3.2). The sender uses a non-blocking protocol since memorization elements (FIFO) are available. These steps end the communication characterization task: all the communications are characterized with their mode, type, resource(s) and protocol(s).

Table 1 Characterization of communication resources

	Width	Throughput	Depth
	Synchronous communication		
Bus	$L_{bus} = L_{data}$	$Th_{i,j} = \frac{V_{data} \times L_{data}}{Max(tr_{i,j}, te_{i,j})}$	-
	Asynchronous communication		
Sender bus	$L_{bus_e} = L_{data}$	$Th^{e}_{i,j} = \frac{V_{data} \times L_{data}}{tr_{i,j}}$	-
FIFO	$L_{FIFO} = L_{data}$	-	$D_{FIFO} = V_{data}$
Receiver bus	$L_{bus_r} = L_{data}$	$Th^{r}_{i,j} = \frac{V_{data} \times L_{data}}{te_{i,j}}$	-

3.2.2. Communication implementation task

The aim of this task is to minimize the number and the size of FIFOs and buses required in the communication

network to support all the data transfers [Gogniat 1997]. Thus, in order to merge communication resources with exclusive life times we use an extended weighted bipartite matching algorithm [Gajski et al. 1992]. Next, data are distributed between internal memories of components according to possible parallel execution of transfers and computations. Control resources (e.g., buffer, multiplexer) necessary to control the communication network are also determined. Then, for each component executing several nodes and edges, the sequences of computation activations and communications are generated. All these operations end the HW/SW integration of the application. Hence, starting from a partitioned and scheduled graph the proposed approach performs the communication synthesis which defines the communication network and the set of controls associated with each component in the final architecture.

4. Design results

To illustrate the principles of this method we consider a frequency domain block adaptative algorithm for acoustic echo cancellation (GMDFα). This application is detailed in Freund et al. [1997b] which expresses also a partitioning and a scheduling of tasks. From these results, an optimized hand crafted approach and the presented method were applied to perform the communication synthesis and the HW/SW integration. Timing measurements of the hand crafted approach result from logic simulations and synthesis. The global and fine evaluation methods have been applied in order to compare both models. Solutions are close since all the transfers are implemented using synchronous communications on a single bus. Moreover, several communications took benefit of the overlapping scheme in order to reduce the global timing overhead cost due to communications. The estimated schedules and the real one provide similar execution times as shown in Table 2.

Table 2 Comparison of final results

	After communication synthesis		
	Fine estimation	Global estimation	Real
Execution time	6683 µs	6722 µs	6800 µs
Communication time	550 µs	700 µs	411 µs
Processor utilization rate	92.7%	92.2%	91%
Hardware unit utilization rate	14.5%	14.4%	14%

However, the communication synthesis with the global evaluation method provides overestimated communication delays since there is a difference of 70% between the estimated solution and the real one. The behavioral model considered in the fine evaluation method provides a more realistic estimation of the communication elapsed times since a difference of 34% is obtained compared to the hand crafted result.

5. Conclusion and future works

From a static partitioned/scheduled graph and a generic underlying architecture, the proposed approach performs communication synthesis and resources optimization in order to provide a dedicated realistic signal processing embedded architecture. An extended communication synthesis method has been defined in order to avoid the manual determination of interfaces between heterogeneous components of the architecture which is error proned and time consuming. On static applications this new method has proven its efficiency and permits a better estimation of the communication schemes implemented in the final architecture. However, this static model may be inadequate to deal with multimedia and telecommunication services that require complex controls. Indeed, the key role of the user interaction and environmental data communications coupled with sophisticated signal processing lead to dynamic execution schemes. Hence, our work will be extended in order to take into account the dynamic features of these applications.

6. References

Filo D., Ku D., Coelho C., De Micheli G. 1993. Interface optimization for concurent systems under timing constraints. *IEEE Transaction on VLSI*, September, pp. 268-281.

Freund L., Dupont D., Israel M., Rousseau F., 1997a. Interface optimization during HW-SW partitioning. *Int. Workshop on HW/SW Codesign*, Braunschweig, March, pp. 75 -78.

Freund L., Israel M., Rousseau F., Bergé J. M., Auguin M., Belleudy C., Gogniat G. 1997b. A Codesign Experiment in Acoustic Echo Cancellation: GMDFα. *ACM Trans. on Design Automation of electronic Systems*, Vol. 2, No. 4, October.

Gajski D., Dutt N., Wu A., Lin S. 1992. HIGH-LEVEL SYNTHESIS: Introduction to Chip and System Design. Kluwer Academic Publisher.

Gogniat G. 1997. Architecture générique et synthèse des communications pour la conception conjointe des systèmes embarqués logicel/matériel. PhD dissertation, Université de Nice Sophia-Antipolis, November.

Gong J., Gajski D., Bakshi S, 1996. Model Refinement for HW-SW Codesign. *Proceeding of European Design & Test Conference*. Paris, March.

Lee E., Messerschmitt D. 1987. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Trans. on Computers*, Vol. C36, No. 1, pp. 24-35.

Yen T.Y., Wolf W. 1995. Communication Synthesis for Distributed Embedded Systems. *Proceedings of International Conference on Computer Aided Design*, San Jose, CA, November, pp 288-294