A multi-granularity data synchronized architecture for HW/SW embedded DSP systems

Michel AUGUIN, Cécile BELLEUDY, Guy GOGNIAT
Laboratoire I3S, 41 Bd Napoléon III
06041 Nice cedex - France.
e-mail: auguin@alto.unice.fr

Yvon JEGOU
IRISA, Campus de Beaulieu
Av Général Leclerc, 35042 Rennes cedex- France.
e-mail: Yvon.Jegou@irisa.fr

1. Introduction

The complexity of embedded systems for telecommunication, multimedia and wireless applications imposes the use of heterogeneous resources (i.e., processor cores, functional units...) in one system. The design methodology of such systems, named hardware/software codesign, must respect cost, performance and time to market constraints. So long as embedded systems were small, handcrafted techniques were sufficient to meet these goals in a reasonable amount of time. However, new embedded systems present such a complexity that it becomes necessary to introduce major improvements in CAD methodologies in order to help designers. These methodologies often allow a hierarchical design, starting from the system design (i.e., mapping the specification into multiple chips) and ending with the technology mapping (i.e., mapping the functional units of one chip to physical implementation). Between these levels an important step concerns the architectural design. The architectural design task, mainly performed by a partitioning of the specification, deals with designing the hardware/software architecture. This partitioning must take the target architecture into account. Hence, to provide high flexibility to this process it is essential to define an efficient model of architecture which includes functionalities adapted to the requirements of the application domain and authorizes an automatic generation of efficient control patterns. This relation between the target architecture and the codesign methodology can be related to the RISC paradigm that links strongly the processor architecture and the associated compiler in order to generate efficient binary code.

2. HW/SW codesign

The main design flow of hardware/software codesign methods consists in modelling the application, partitioning this model to define specifications of hardware and software units, constrained code generation for the software unit, constrained communication and system synthesis. Moreover, hardware and software performance/ cost estimators are needed to guide the partitioning step. Generally, HW/SW published codesign techniques address only a subset of these operations due to the intrinsic complexity of them.
The Siera framework [10] focuses on rapid prototyping of multi-board hardware/software systems. The template architecture is composed of different layers that correspond to this multi-board level design. Since we address the design of ASIC chips with processor cores and dedicated cells this multi-layered template is not convenient for our purpose. The Ptolemy framework [1] allows to model and simulate applications with data flow graphs and performs code generation. In [5] a HW/SW partitioning algorithm is presented for the task level of the data flow model but while Ptolemy is able to generate code for a multiprocessor system, this partitioning algorithm considers an architecture with only one processor. In [2],[3] HW/SW codesign methods deal with C based representations extended with communication process facilities. These codesign methods consider an underlying architecture with only one processor whereas RISC (possibly with multimedia instructions) and DSP cores may be integrated into one chip with specific hardware cells to form a dedicated heterogeneous bi-processor system.

3. Characteristics of target applications

It is a very hard problem to define an efficient method dealing with a broad spectrum of applications. Better results are obtained if a restricted application area is considered. Hence, main characteristics of targeted applications that have to be handled by a codesign method, must be pointed out. The important characteristics we consider in embedded signal processing telecommunication applications are:

1. At the upper level they are modelled by functions connected in a data flow style.
2. This description contains an intrinsic parallelism between tasks.
3. Functions may have predefined implementations either with code sections on a processor or off-the-shelf hardware cells.
4. But some functions may have no a priori implementations. These functions are hierarchically detailed and the lowest levels have generally a

control oriented description (C or C++ for example). These hierarchical levels determine various grains of computations.

5. Furthermore, image and frequency domain processing leads to array or vector computations and communications whereas time domain processing of mono-dimensional signals implies scalar computations and communications.
6. Numerous applications have a static behavior that means that order of computations does not depend on data values [6].

Consequently, a multi-formalism model may be required according to the granularity of computations. The Ptolemy flow graph approach is an example of a mixed data-flow/control oriented modelization of applications. In the same way, in Siera the system representation is formulated with a network of asynchronous processes expressed using a procedural or an applicative language according to the nature of the process.

The aim of a codesign methodology is to map a specification on a HW/SW system that matches some performance and cost constraints. The HW/SW system is deduced from a generic model that must be suited to characteristics of applications in order to facilitate constraint meetings. In the following section a generic architecture for HW/SW codesign adapted to the above characteristics is presented.

4. HW/SW partitioning of a specification

Codesign methods begin generally with a hardware/software partitioning of the tasks with the aim of meeting cost and performance constraints. Partitioning methods consider an underlying scheduling of tasks through timing evaluations or are based on scheduling techniques. Hence, the partitioning sets out the execution scheme of tasks that may contain some parallelism. Parallel execution of tasks with predefined implementations imposes to instanciate corresponding FUs in the architecture. Tasks with a control oriented description may be modelled as a set of threads with precedence relations. Parallel execution of these tasks imposes parallel execution of threads (Fig. 1). Therefore, the generic architecture must be

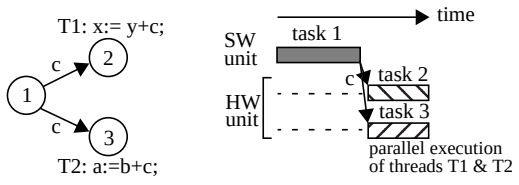

Fig. 1 Parallel execution of threads

able to support this parallel execution scheme.

5. A generic architecture for codesign

Our generic model for HW/SW codesign is based on the DSPA (data synchronized parallel architecture) architecture [4] which was developed for high performance scientific computing. The initial DSPA model is depicted in [4]. Each functional unit (ALU, Mul, Mem,...) receives its instruction flow through a FIFO from a VLIW type instruction memory. An instruction is executed by a functional unit (FU) as soon as data are available on its input ports. Outputs and inputs of FUs are connected through an incomplete crossbar network. Each crosspoint corresponds to a FIFO. A sequencing unit computes the address of the next instruction. This model presents attractive characteristics:

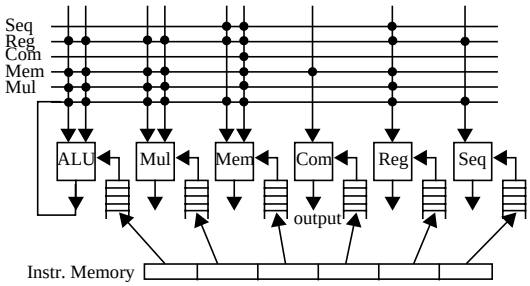

Fig. 2 The basic model of DSPA

1. FUs may be asynchronous since they are synchronized on data arrivals. Then, communications between FUs are asynchronous. The Siera template architecture [10] considers also this approach to avoid the distribution of a global clock over the whole system.
1. Software pipelining techniques can be applied to speed-up computations on regular structures of data.
2. FUs are controlled and interconnected with the same scheme, including communication ports (Com). This facility allows to consider heterogeneous FUs in the same structure: for example DSP processors, fine grain or coarse grain dedicated FUs.
3. Adding FUs in the structure is easy since the network structure and the control mechanism are graceful extensible. Consequently, a specific system can be constructed from the DSPA generic model with a synthesis mechanism.

The DSPA model is a good candidate for codesign methodologies but the important use of FIFOs constitutes a major drawback for embedded system design. Therefore, refinements and improvements to the basic model are required to get a real generic architecture for HW/SW codesign.

5.1 Communication model

Crosspoint FIFO queues result from the asynchronous transfer model between FUs. This communi-

cation model leads to an ASAP (As Soon As Possible) style execution 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 to a synchronous transfer mode without overstepping timing constraints. Each crosspoint of the architecture can handle synchronous and asynchronous transfer modes. The synchronous transfer mode is supported with a *rendez-vous* mechanism. The asynchronous transfer mode is based on FIFO, hence the protocol associated can be blocking or non blocking. The protocol is blocking if before reading or writing into a crosspoint the corresponding unit must check the availabilities of data. If non blocking no verification needs to be done before writing or reading data from a FIFO. The communication structure of a crosspoint in represented in Fig. 3. All the communication and control signals supporting data transfers are defined. If a crosspoint needs to handle synchronous and asynchronous transfer modes, then the asyn-

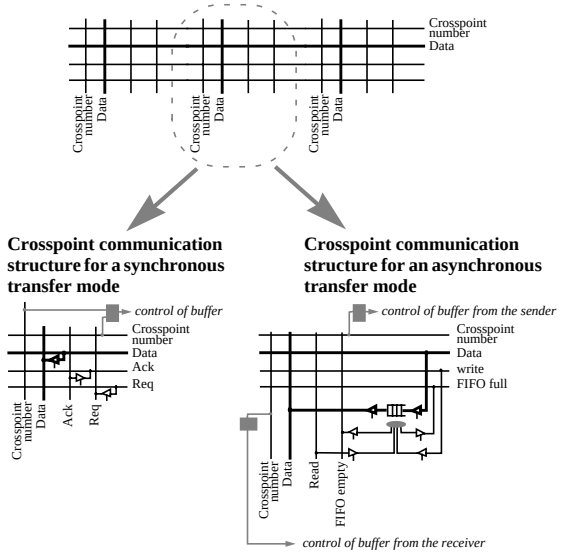

Fig. 3 Crosspoint communication structure of the architecture

chronous communication structure is used for both types of transfers. Note that hardware resources needed for asynchronous transfer modes are more costly than for synchronous transfer modes. Thus, the schedule of the application must minimize that type of communication.

5.2 Generic model of HW and SW units

A system is composed of interconnected HW and SW units. A system design automation is facilitated if these units have a common interface. So, each functional unit of the application uses, from the communication network point of view, the same interface. This interface deals with controls and communications. Control and communication signals are used in order to support synchronous and asynchronous communication protocols as shown in Fig. 3. The communication network only deals with these signals. The kernel has several models corresponding to different types of resources. For example, in the case of a dedicated functional unit like a FFT unit, the kernel is composed of the instruction memory, the computation cell and controllers. In the case of a DSP the kernel has also data and program memories dedicated to the processor. These memories are included in the kernel in order to ensure fast and simple memory access. The resources supported with this type of kernel are specific functional units, logic functions, memory elements (registers, memories) and processors. Each of these kernel must respect the network access model in order to ensure a correct execution of the application. Hence, even if the kernel is specific to each unit to ensure computation more efficiently, input and output signals of interfaces are standards

5.3 Control flow of the model

The partitioning and scheduling tasks of the application provide the instruction set of each unit composing the architecture. Each set of instructions represents all the operations computed on the associated unit. An instruction contains the information required to execute the operation. In the case of dedicated functional units, these information are: the location of the input and output data crosspoint in the communication network. When the unit can support several computation schemes (for example, addition and substraction), then the instruction model also specifies the operation type. In order to ensure correct application execution each unit has two controllers. A general con-

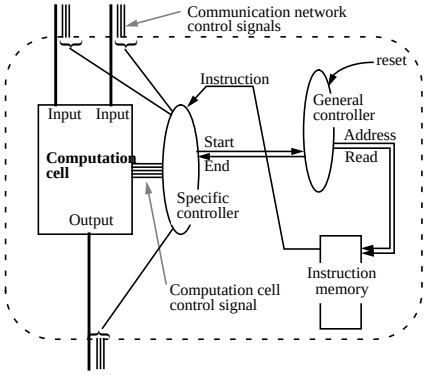

Fig. 4 Control structure of functional units

troller provides the data control oriented scheme of the architecture. This controller supports access to instructions, control flow execution and activation of a specific controller. The specific

controller decodes instructions and ensures adapted communication protocols in order to send and receive data from the communication network. The specific controller can support several communication protocols if required in the data transfer network. The control of the computation cell is also managed by this controller (Fig. 4). When the computation cell is a processor core, the specific controller activate the execution using an interruption mechanism. All the functional units have their own clock in order to take advantage of the application parallelism. Note that computation cell can be defined at several levels of granularity.

5.4 Mixed granularity levels

As mentioned in section 3. parts of an application may be represented with a coarse grain data flow style whereas other parts are modelled with a fine grain control oriented language (C or C++). The fine grain imposes the synthesis of a specific system. Furthermore, since the coarse grain part corresponds to a task level representation, the sharing of FUs by different tasks may be limited and the system may be oversized. Therefore, some tasks have to be modelled at a lower level of granularity. In this case, high level or behavioral synthesis [7]can be applied if computations are not too complex (without array processing for example) otherwise more elaborated techniques are preferable [11], [8]. The two approaches are considered since they lead to the design of an FU structure integrated in the generic architecture. Since the generic architecture is adapted to synthesis we use this model for fine grain synthesis. Currently, we are adapting a synthesis technique[8] for addressing this template architecture. But as it is illustrated in Fig. 1, some threads must have a parallel execution in the hardware unit. Hence, to avoid a total replication of resources the sharing of FUs by parallel threads is enabled in the generic architecture. This is achieved by tagging data computed by e thread with an identification number of the thread. Availability of data on input ports of the FU allows to select the FSM according to the thread to be executed (Fig. 5). This sharing of resources may be performed after synthesis of threads associated with tasks.

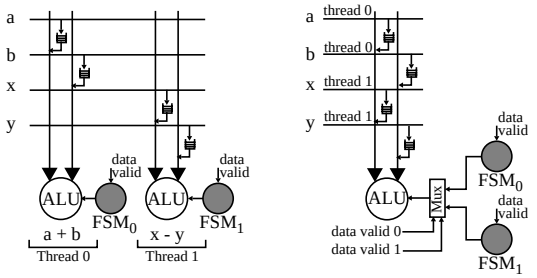

Fig. 5 Sharing of resources by threads

6. GMDFα architecture example.

To illustrate the principles of this generic architecture we consider a frequency domain block adaptative algorithm for acoustic echo cancellation (GMDFα). The flow graph is depicted in Fig. 6. The precedence constraints between two nodes

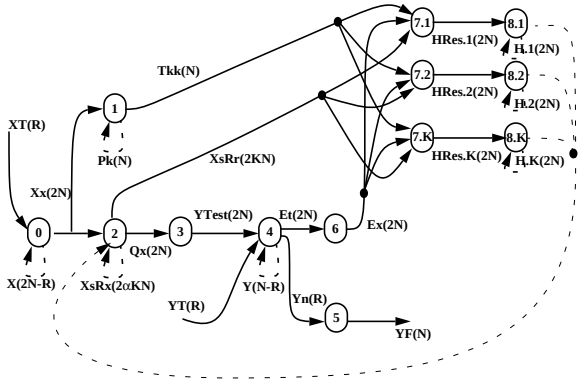

Fig. 6 Flow graph of GMDFα

are expressed by an arc. On each arc, the name and the volume of data are given. A dotted line means that this data will be used at the next iteration of the algorithm.

SW Nodes	HW Nodes	Scheduling 56002 begin -> end (µs)	Hardware operators
	0	0 -> 86	FFT
1		87 -> 558	
2		559 -> 1288	
3		1289 -> 1758	
4		1759 -> 1820	
5		1821 -> 1854	
	6	1821 -> 1907	FFT
7.1 7.2...7.8		1908 -> 2455 2456->...->5743	
	8.8	5744 -> 5840	FFT, Adder
		5840 µs	FFT, Adder

Table 1 Partitioning of GMDFα

Node 0 computes FFT on the input signal (XT). Node 1 is the normalization block. The output of node 2 gives the estimated output in the frequency domain (Qx which denotes arrays Qr and Qi of complex values) by a convolution product, and after an FFT^{-1} (node 3), we obtain the estimated output in the time domain (Ytest). The difference between the desired output (YT) and the estimated output is calculated in node 4. Node 5 provides the echo (YF). Node 6 transforms the error signal (Et) into an error signal in the frequency domain (Ex) by one FFT. In nodes 7.i and 8.i, (i=1..k) filter coefficients are computed by FFT and FFT^{-1} operations of each elementary cells. In Table 1 is

given a HW/SW partitioning of the flow graph [9]. This partitioning is defined through performance and area estimations for each node of the application. The software unit is a DSP56002 processor and the hardware part is composed of a FFT operator and an adder.

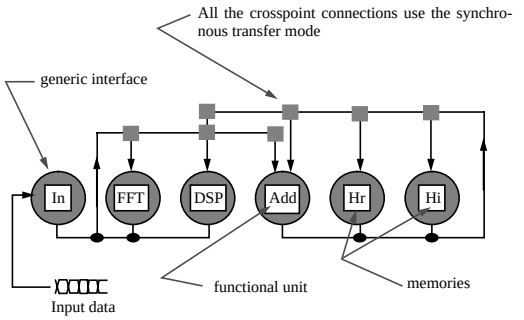

Fig. 7 HW/SW architecture for GMDFα.

In the initial architecture all the units are connected through a crossbar network (six buses). The schedule of application nodes allows to use only the synchronous transfer mode. Moreover the network optimization allows to reduce the number of buses to only two. The hardware communication resources of the architecture for that application do not increase significantly the area. Thanks to the optimization we benefit of the efficiency of the architecture model and we provide a small communication network. We avoid the high cost due to FIFO structure. This distributed control architecture permits to exploit the parallelism of the application according to data arrivals. Specification constraints in term of performance and area are met with this implementation. These preliminary results show that this model is a good generic candidate architecture for implementation of efficient HW/SW systems.

7. Conclusion and future works

In this paper a generic architecture for HW/SW codesign methodologies is presented. Main characteristics of this model include asynchronous behaviors of functional units with synchronizations on data arrivals, common encapsulation of functional units that allows easy extension capabilities and integration of heterogeneous HW/SW units with various granularities, functional units can result from the synthesis of fine grain tasks using this generic model, sharing of resources between parallel threads is enabled. But the communication model can lead to excessive communication resources, therefore partitioning methods that consider transfer operations and optimization heuristics that transform asynchronous communications into synchronous ones are areas for future works.

8. Acknowledgments

We would like to thank Professor M. Israel, Doctor F. Rousseau and Doctor L. Freund from The LAMI, Université d'Evry and J. M. Berge from France Telecom - CNET for all the fruitfully conversation we had with them.

9. References

[1] BUCK J., HA S., LEE E.A., MESSERSCHMITT D.G. Ptolemy: a framework for simulating and prototyping heterogeneous systems. *Int. Journal of Computer Simulation: special issue on Simulation Software Development.* 4 april, 1994.

[2] ERNST R., HENKEL J., BENNER T. Hardware-Software Cosynthesis for Microcontrollers. *IEEE Journal Design and Test of Computers.* 64-75, december, 1993.

[3] GUPTA R.K., DE MICHELI G. Hardware-Software Cosynthesis for Digital Systems. *IEEE Journal Design and Test of Computers.* 29-41, september, 1993.

[4] JEGOU Y., SEZNEC A. *Data synchronized pipeline architecture: pipelining in multiprocessor environments.* Technical Report 503, INRIA/IRISA, Mars, 1986.

[5] KALAVADE A., LEE E.A. Hardware/software co-design using Ptolemy - A case study. *Proceedings IFIP International Workshop on Hardware/Software Co-Design*. Grassau, Germany, May 19-21, 1992.

[6] LEE E., MESSERSCHMITT D. Synchronous data flow. *Proceedings of IEEE.* 75(9):1235-1245, September, 1987.

[7] McFARLAND M.C., PARKER A.C., CAMPOSANO R. The high level synthesis of digital systems. *Proceedings of IEEE.* 78(2):301-318, february, 1990.

[8] MENEZ G., AUGUIN M., BOERI F., CARRIERE C. A partitioning algorithm for system level synthesis. *Proceedings ICCAD92*, pages 482-487. Santa-Clara, California, november 8-12, 1992.

[9] ROUSSEAU F., BENZAKKI J., BERGE J.M., ISRAEL M. Adaptation of force-directed scheduling for hardware/software partitioning. *Proceedings Int. Workshop on Rapid System Prototyping*. Chapel Hill, NC, June 7-9, 1995.

[10]SRIVASTA M.B., BRODERSEN R.W.,. SIERA: A Unified Framework for Rapid-Prototyping of System-Level Hardware and Software. *IEEE Transactions on Computer-Aided Design.* 14(6):676-693, June, 1995.

[11]WILBERG J., CAMPOSANO R., ROSENSTIEL W. Design flow for hardware/software cosynthesis of a video compression system. *Proceedings Int. Workshop on Hardware-Software Co-Design*, pages 73-80. Grenoble, France., September 22-24, 1994.