An Object-Based Executable Model for Simulation of Real-Time Hw/Sw Systems

O. Pasquier, J.P. Calvez
IRESTE, University of Nantes, FRANCE, email : olivier.pasquier@ireste.fr

Abstract

This paper describes a simulation technique for Real-Time Hw/Sw systems based on an object executable model. It allows designers to seamlessly estimate and verify their solutions from a high-level functional description to a Hw/Sw partitioned design. The same model, enhanced with algorithms, can be used to simulate interpreted models in order to observe the behavior of a whole system and its environment, as well as uninterpreted models which are useful for performance estimation and so to help Hw/Sw partitioning. Our (co-)simulation technique is based on the translation of a high level model of the application into a C++ program which uses a library of predefined classes. While running, the program produces an event trace which contents can be set by the user. This technique allows to quickly analyze the influence of application or architecture parameters on the application behavior.

1: Motivations and Goals

The top-down design of any embedded system needs solutions to be verified and analyzed during each step and as soon as possible. Since systems are correctly verified only when they are embedded in their final environment, an abstract but realistic model of the system environment has to be developed early in the design process. This model is also needful for system specification. Then, the functional design phase leads to develop a functional and technology-independent model of the system. The refinement technique enables designers to start from a high-level functional model and then gradually replace the behavior of the function by a more detailed structural description.The detailed Hw/Sw solution is then the result of a partitioning work based on estimations extracted from a performance model, and decisions related to non-functional constraints.

Our objective is to provide a tool to validate solutions at each step of the design process. The quality and efficiency of the design and verification process highly depends on extra work it requires. Thus the design process has to be seamless and verification must be based on automatic tools.

This work is based on the MCSE methodology which is a complete design process for real-time systems and also useful for CoDesign [2].

2: Description model and simulation technique

Different simulation techniques have been developed to cosimulate Hw/Sw applications. They often require a model of the application processor which may not be available in an early stage of the design process [1]. Our solution, based on the MCSE description model, overcomes this difficulty by considering applications at a higher abstraction level.

2.1: The MCSE description model for co-simulation

The MCSE model, used to describe Hw/Sw systems, includes the functional model, the executive model and the mapping between both models [2][3].

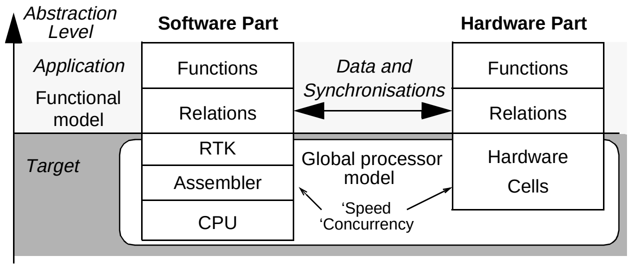
The functional model describes the organization of the system and its environment as well as the behavior of internal and external functions. Functions are linked through three kinds of relationship : event, shared-variable, port. The behavioral model is a partial-ordered set of operations, conditions (data input or synchronization) and actions (data output or signal). At an abstract description level, the model is uninterpreted : operations or in/out actions are modeled by an execution time attribute. When algorithmic details are known, they are added to the description of operations, but timing simulation is still based on time attributes. The model then becomes interpreted. Different attributes are used to define each basic element of the description model (concurrency, capacity, access time, ...).

The functional model is hierarchical. It means that a function can be first described by a rough behavioral model and then gradually refined by a structural model. This allows incremental validation of the solution.

For co-simulation the model must include functional and hardware architectures and the partitioning result (mapping). In MCSE, the executive model represents the hardware architecture. This model is graphically similar to the functional one but objects have a different meaning : processor, signal, shared-memory, communication node. Each component is also specified by attributes (scheduling policy, power, concurrency, ...).

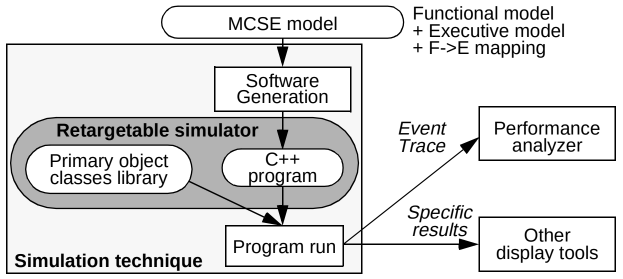
2.2: The simulation technique

According to the multi-layer model given in Figure 1, the upper-level behavior of an application (functions and relationships) can be analyzed without considering Hw/Sw partitioning. Then, when considering the hardware target after partitioning, our global model means that "hardware" functions can process concurrently, whereas on software processors, only one function can run at the same time, according to a Real Time Kernel (RTK) scheduling policy. This is defined by the attribute *concurrency*.



-Figure 1 - Abstract layers of Hw/Sw systems.

Our simulation process is depicted in Figure 2. Systems with their environment are graphically described, according to the MCSE model, with graphical description tools. A software generator then automatically translates this model into a C++ program. Each object of this program implements a component (function, processor, relation) of the system model as an instance of a class which inherits from a "primary class" defined in the *object library*. A "primary class" implements the semantics of a component of the MCSE model: active components (functions and processors), relation components. These components implement parallelism, condition synchronization and mutual exclusion like in [4].



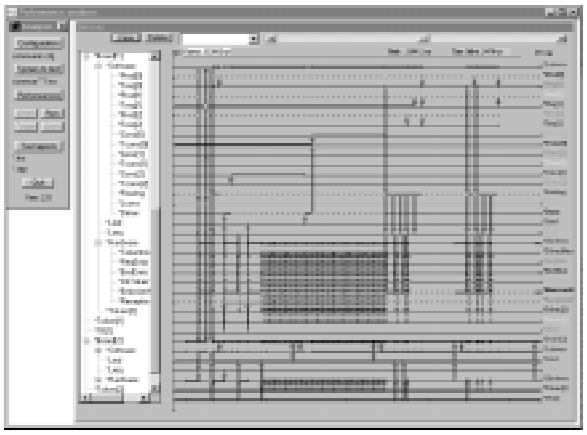
-Figure 2 - Our co-simulation technique.

Four primary classes have been defined for the components of a structural (functional and executive) view :
- Active Element: for functions and processors.
- Event: for function and processor synchronization.
- SharedData: for shared variables and shared memories.
- Port: for message transfer and communication node.

Attributes are used to differentiate elements at functional and architectural levels. Similarly, classes have been defined for basic elements of the behavioral description model.

To implement interpreted simulation, we have replaced the simulation of each CPU instruction (like in Ptolemy or for CVE) by the direct execution of the algorithm of each behavior operation and a simulation of their execution time. Of course, result accuracy is different, but it is fair enough for partitioning and for first high-level verifications.

The generated C++ program fully respects the hierarchy of the application model (hierarchy of objects). Designers can thus easily enhance it by any sort of instructions. The program run produces a sequence of events. Each event represents a significant internal state change, for example wait for processor availability, shared variable access, etc. Events are processed and graphically displayed to show performance results (figure 3 for example). If more specific results are needed, designers can enhance the object code to produce them.



-Figure 3 - Example of time line display.

3: Conclusions

In this paper we have described an efficient high level co-simulation technique for system-level models. Specifications and properties of solutions for the designers' point of view are expressed by the MCSE model which enables to seamlessly describe systems before and after Hw/Sw partitioning. Then the simulation is realized by a C++ program automatically generated from design results.

These latter ones are powerful for system and architecture performance modeling, estimation and evaluation. The resulting tool is directly integrated as a support for the system design and CoDesign MCSE methodology, but can also be used independently of MCSE by writing models directly in C++.

References

[1] J.T. Buck, S. Ha, E.E. Lee, D.G. Messerschmitt, Ptolemy: a framework for simulating and prototyping heterogeneous systems, Int. Journal of Computer Simulation, Vol 4, April 1994, pp. 155-182, 1994, pp187-212

[2] J.P. Calvez, Embedded Real-time Systems. A specification and Design Methodology, John Wiley, 1993

[3] J.P. Calvez, A System-level performance model and method, "Current Issues In Electronic Modeling", Issue #6: Meta-modeling: Performance, Software and Information Modeling, Kluwer Academic Publishers, 1996, pp 57-102

[4] B. Kienhuis, E. Deprettere, et al., The Construction of a Retargetable Simulator for an Architecture Template, Proc. of the 6th Int. Workshop on Hardware/Software Codesign, Seattle, Washington, 15-18 March 1998, pp 125-129.