THE SUCCESSIVE REFINEMENT METHODOLOGY IN THE EMBEDDED SYSTEMS DESIGN

Sandro Renato Dias, Diógenes Cecílio da Silva Júnior
Departamento de Engenharia Elétrica – Universidade Federal de Minas Gerais, Brazil
`{sandrord, diogenes}@cpdee.ufmg.br`

ABSTRACT

The methodology of successive refinement is becoming a strong point in the project of embedded systems. Here we will present the methodology and to propose a refinement model, defining levels and refinement rules to each level, enabling to the designer to develop his projects with more agility and possibility of verification and simulation all moment.

1. INTRODUCTION

The use of the Successive Refinement Methodology is crescent and several languages, methodologies and tools were developed or were adapted to facilitate the use of this methodology. Its main aim is to work with different abstraction levels to facilitate and speed the design development. It is possible with the use of hardware description languages based on C/C++ (such as SystemC), since they facilitate the hardware-software co-design, i.e., the development of the software without a finished physical stable hardware, before the end of the project.

Usually 50% of the debugging time is spent with the specification incorrectly translated or ambiguous, such problems are easily solved using refinements since the designer can use the same language from the specification to the synthesis level. This work aims to propose a model of refinement with rules to define the levels.

2. SUCCESSIVE REFINEMENT METHODOLOGY

Abstraction is a powerful technique for the design and implementation of complex hardware systems. It allows us to deal with the complexity of the system hiding unnecessary details until later use. The different quantity of details means different levels of abstraction, Müller, Rosenstiel and Ruf [1]. The semantics of the modeling based on the abstraction is well described by Gerstlauer and Gajski [2], where they demonstrate the importance and advantages of its use. Abdi and Gajski [3] developed the Model Algebra, a mathematics formalism which can be used to represent system models and its transformations in the modeling levels using refinements. The consequences of the level abstraction increasing methodology, showed by Schutten [4], are the reduction of the need of SoC verification, reduction of design time, the gain in better obtaining results and the possibility of anticipates the concurrent hardware and software development.

Using SystemC language, the project doesn't need to be converted, by a great effort, from a description in C to a hardware description language, like Verilog. Instead of that, the project is refined in small sections to add hardware and timing constructs towards the system final objective. Using the successive refinement methodology, the designer can easily implements changes in the project and detects and solve problems during the refinement.

3. MODELING LEVELS

A level hierarchy applicable to digital system projects was firstly described by Bell and Newell, 1971, (apud MacFarland, Parker and Camposano [5]), and was used specifically to computational systems. A view of this hierarchy was proposed by MacFarland, Parker and Camposano, 1990, like an update of the Bell and Newell previous work. Gong, Gajski and Bakshi [6] describe a refinement model to the simultaneous development of hardware-software. This model uses the SpecCharts language to describe the modeling and its refinements. Bhasker [7] presents a design flow based on SystemC. The levels are well defined, consisting on four models (System Level Model, Timed Model, Behavior Level Model and RTL Model) that aggregate a different feature to the design in each refinement.

The development of the work presented here is a model whose levels are detailed below. This definition of levels is necessary for demarcation of the purpose of the work. The refinement rules and implementation decisions here presented were compiled from [6], [7], [8], [9], [10], [11], [12] and [13].

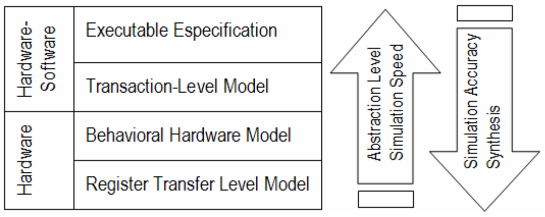

Figure 1 – Refinement levels and their relation with simulation speed and accuracy

3.1. Executable Specification Level Model

The executable specification represents the straight translation of the design specification in SystemC, so that it is completely independent of the target implementation. This level does not implement time delay and the communication between modules is point-to-point. The data types are also abstracts. Because of the high abstraction of this level and by the absence of specificity in the code regarding the final objective of the project (cycle, pin and time precision absence) this level is not able to, therefore, be synthesized.

The implementation decisions for this level are: definition of the process Thread, with the modules implementing the specific functionalities of the system; the use of the channel sc_fifo for communication between the modules, with the use of the directives read and write for, respectively, reading and writing in the fifo; initialization of the simulation with the sc_start(-1).

3.2. Transaction-Level Model (TLM)

In this level, time delays are added to the process inside the project to reflect the time constraints of the specification, target implementation processing and latencies of specific implementations. The communication between the modules is modeled using function calls; it has accuracy in terms of functionality and approximation in terms of time. However, it is not modeled in a precise structure form regarding to the target implementation, therefore details of communication (protocol used, by example) are treated subsequently.

Refinement rules for the TLM: to insert wait in each computation and in each communication inside the processes; to change the sc_start for a time defined of simulation; to allocate modules for represent processing elements taking care with the processes inside them; to map global variables in local memories of the elements or in shared memory; to refine sc_fifo in global channels; to group global channels in hierarchical channels to form buses; to use interfaces to access the bus; to refine the accesses to the processes ports, from the previous level, to function calls that implements the interfaces of access to the bus.

3.3. Behavioral Hardware Model (BHM)

The BHM is pin and cycle accurate and its internal structure is closest to, but does not necessarily reflect, the target implementation. This model is synthesizable, despite still in an algorithm level, to show the inputs and outputs of the design with clock cycle accuracy of the final system. Beyond that its simulation is quicker than the subsequent level, RTL, due to its high level of abstraction.

The refinement rules for the BHM are: to use functions for manage the complexity of the system, if necessary; to replace the structures not synthesizable by synthesizable ones; to insert the cycle precision through waits in the operations of input and output; to insert adaptors in order to implement a bus in the RTL level. Insert these adaptors in the processing elements; to replace processes sc_thread by sc_cthread; to include a port clock in each module.

3.4. Register Transfer Level Model (RTL)

The RTL model is the lowest abstraction level supported by SystemC and corresponds to the digital hardware synchronized by clock signs. It describes the behavior of the finite states machine of the final project and its datapath. It is pin and clock cycle accurate, where each operation is carried out should have its defined precision. In this model, all the communication between processes occurs through signs. Its structure reflects precisely the registers of the target implementation and the combinational logic between them.

The refinement rules for the RTL model are: to separate control logic and datapath, replacing sc_cthread by sc_method; to decide the datapath architecture, replacing variables by signs; to define initial values of variables and signs in the constructor of the module; to define an explicit finite states machine for the control logic.

4. RELATION BETWEEN THE LEVELS

Higher abstraction levels imply higher simulation speed. This because a higher abstraction implies in less hardware implementation details to be described (Table 1), generating a simpler code and a quicker execution. In compensation, lowest level, as much as closest to the level RTL, higher will be the precision of the simulation and the capacity of synthesis, however, the simulation becomes more delayed due to the level of details necessarily employed. These details increase each refinement, towards to the RTL.

5. CONCLUSION

The increase of the complexity and the increasingly scale integration in the present days lead the designers to redefine their methodologies of design. The use of a language that enables diverse levels of abstraction beyond a coherent and successive refinement permits the designer more liberty during the development. Besides, the possibility of a simultaneous development of hardware and software reduces the total time of the project, permitting the software to be developed over an environment implemented in a level of abstraction that will simulate the final hardware. We believe that SystemC is a powerful language for the designer and the use of the Successive Refinement Methodology will increase the productivity of the designer, facilitating the project of systems more and more complex. The levels are sufficient to help the designer to develop embedded systems more easily because they are essential to the refinement use. They add the specific feature at each refinement towards the target implementation. The use of a TLM level instead of a single Timed Model gives the correct attention to this feature at the correct time.

Accuracy Metrics	ESM	TLM	BHM	RTLM
Functional	Yes	Yes	Yes	Yes
Computation Timing	No	Approx	Approx	Cycle
Communication Timing	No	Approx	Cycle	Cycle
Communication Protocol	No	Approx	Exact	Exact
Structural	No	Approx	Approx	Exact
Pin	No	No	Yes	Yes

Table 1 – Level accuracy in SystemC, adapted from [10]

6. REFERENCES

[1] Müller, W., Rosenstiel, W., Ruf, J. *SystemC – Methodologies and Applications*. Kluwer Academic Publishers, USA, 2003.

[2] Gerstlauer, A., Gajski, D. *System-Level Abstraction Semantics*, Center for Embedded Computer Systems, University of California.

[3] Abdi, S., Gajski, D. *Provably Correct Architecture Refinement*. Technical Report CECS-03-29. September 30, 2003. Center for Embedded Computer Systems, University of California, Irvine, USA.

[4] Schutten, R. *Raising the Level of Abstraction Reduces System-on-Chip Verification*, March 2004, Synopsys, Inc.

[5] McFarland, M., Parker, A., Camposano, R. "The high-level synthesis of digital systems". Proceedings of the IEEE, special issue on *The future of computer-aided design*, Vol. 78, No. 2, pp. 301-318, Feb. 1990.

[6] Gong, J., Gajski, D., Bakshi, S. *Model Refinement of Hardware-Software Codesign.* University of California, Irvine, ACM Transactions on Design automation of Electronic Systems, Vol. 2, No. 1, pp. 22-41, Jan. 1997.

[7] Bhasker, J. *A SystemC Primer.* Star Galaxy Publishing, 2002.

[8] Dias, S. *Proposta de uma ferramenta de apoio ao projetista de sistemas embutidos usando SystemC*. 1ª Semana de Pós-Graduação do CPDEE/UFMG, setembro, 2005.

[9] Sypnopsys Inc. *CoCentric SystemC Compiler Behavioral Modeling Guide.* Version 2000.11-SCC1, March 2001.

[10] Walstrom, R. *System Level Design Refinement using SystemC.* Master thesis. Iowa State University, Iowa, 2004.

[11] Grötker, T., Liao, S., Martin, G., Swan, S. *System Design with SystemC.* Kluwer Academic Publishers – Boston/Dordrecht/London, 2002.

[12] Open SystemC Initiative. *SystemC Version 2.0 User's Guide – Update for SystemC 2.0.1.* 2002.

[13] Sypnopsys Inc. *CoCentric SystemC Compiler RTL User and Modeling Guide.* Version U-2003.06, June 2003.