.NET Framework - A solution for the next generation tools for system-level modeling and simulation

J. Lapalme[(1)], E.M. Aboulhamid[(1)], G. Nicolescu[(2)], L. Charest[(1)], F.R. Boyer[(2)], J.P David[(1)], G. Bois[(2)]
(1) DIRO, Université de Montréal *(2) DGI, Ecole Polytechnique de Montréal*
http://www.esys-net.org

1. Introduction

Nowadays the use of System Level Description languages is mandatory for the efficient design of complex systems. These description languages are exemplified by SystemC and SystemVerilog. SystemC is meant for hw/sw system level design. It provides all the basic concepts used by HDLs (e.g. modules, ports, signals, time) and more abstract concepts (e.g. interfaces, communication channels, events). Most of the features for software modeling are not completely implemented: dynamic process creation, process control (e.g. suspend, resume, kill), preemption, software specific communication primitives (e.g. monitors, semaphores). Connecting tools to SystemC seems a relatively costly task due to the complexity of C++ as well as the lack of introspection mechanisms. SystemVerilog is an extension of Verilog, adding a significant set of features including high-level concepts for abstract system modeling and simulation, testbench automation, and the integration of Verilog with the C language. While SystemC is an "open" environment, based on C++, SystemVerilog is a standalone language.

Despite all these efforts, all the following mandatory requirements for an efficient modeling and simulation framework are still not provided by a single existing environment: (a) easier software components specification and their integration into an overall hw/sw system specification; (b) clean programming features and memory management to enable less error prone models, easier specification and reuse; (c) introspection and the possibility of annotating models which simplifies the debugging and analysis of complex specifications as well as the development of synthesis and verification tools; (d) translation to a standard intermediate format to enable the design of EDA tools independently of the used description languages; (e) integration in a distributed web-based design environment and easy system documentation to facilitate cooperation between different designers groups and allow remote processing; (g) multi-platform and multi-language features for describing and designing the overall embedded systems composed of heterogeneous components; (h) openness of the system.

Most of the presented requirements are already provided by the .NET Framework (announced by Microsoft in 2000 and which led to three ISO standards in April 2003). It is intended to be platform and operating system independent. It simplifies application development in the highly distributed Internet environment. It is meant to facilitate the reuse of existing code. Currently, the Microsoft version supports 20 languages (e.g. Java, C++, C#). These languages are mapped to a Common Intermediate Language and execute in a Common Language Runtime. This simplifies programming by assisting with different tasks including memory management and error handling. .NET provides also a class library that can be used by the developers to extend the capabilities of their software.

We propose a new .NET Framework based system level modeling and simulation environment called Esys.NET (Embedded Systems Design with .NET). This environment respects all the requirements above. In particular, it allows (1) cooperation – by enabling web-based design and multi-language features, (2) easy systems specification task – by enabling integration of software components running application and operating systems and by alleviating memory management, (3) link to automatic refinement tools – by enabling translation of specification models into a standard intermediate format and annotation of specification models, and (4) comparative performances with existing environments.

2. Modeling and simulation in ESys.NET

The core of ESys.NET was developed on top of .NET using C#, a strongly typed object-oriented language designed to give the optimum blend of simplicity, expressiveness and performance. In terms of implementation, ESys.NET is based on a set of classes encapsulating the concepts of modules, communications and events. One of its important characteristics is that it offers the designers the possibility to easily specify execution directives by tagging the different concepts in the specification. This allows the association of a thread or parallel method semantic to a class method, the addition of a sensitivity list for a parallel method or a thread, the call of methods before/after the execution of a process, the execution of a class method at a specific moment during the execution. This was implemented by exploiting attribute programming provided by .NET and C#. The available directives also permit the hooking of external tools to simulator and model.

Figure 1 illustrates the ESys.NET layered architecture.

1530-1591/04 $20.00 (c) 2004 IEEE

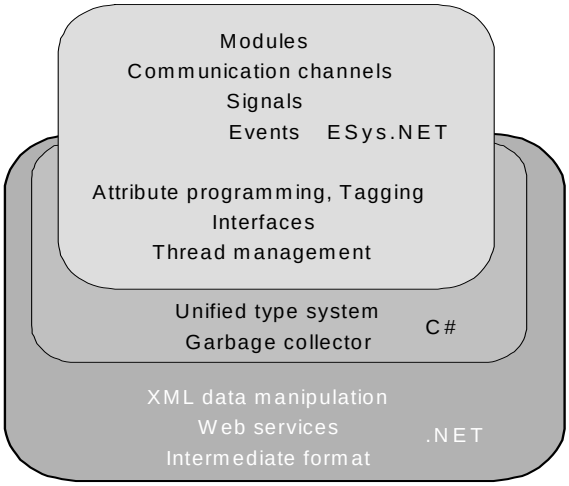

Figure 1: Layered architecture of ESys.NET

To prove the efficiency of ESys.NET we performed several experiments. The main criteria that we used for the evaluation were the performance and the applicability for concrete systems modeling and simulation. We firstly compared the performance of the C# language to the C++ language using the simulation model of a DLX processor. We measured the simulation time of this application for the C# specification execution on .NET and the C++ description executing natively and we obtained that the two languages present comparable capabilities in terms of simulation speed – the C# execution time penalty was below 10%. In addition, we modeled and simulated a second concrete application. In order to compare with SystemC, we used an application provided by SystemC. The application consists of six components interconnected via a communication channel. These components may be masters or slaves of the channel: a master module requires communication primitives from the channel and the slave module offers services to the communication channel. The six components of the system are:

- the fast memory and the slow memory differing by the number of clock cycles necessary to read/ write data; they are slave modules of the communication channel;
- three master modules reading/writing data to/from the memories;
- an arbiter providing a priority based management for the concurrent requests from the masters.

For this system we verified the correctness and the performances of the simulation by comparing the results in our environment with those given by SystemC.

3. Current status and future work

ESys.NET currently provides most of the concepts found in the SystemC environment. It adds, among other concepts, preemptive multithreading existing in .NET and all the operating system primitives planned in SystemC 3.0. ESys.NET is intended to be an evolution of SystemC by offering a user-friendlier environment free of macros, pointers, function prototyping, etc. It is also intended to be a superset of SystemC's core functionalities extending it with features like automatic memory management, system level primitives, strong typing, native interfaces, safe pointers, reflective capabilities, remoting, dynamic thread creation/control. Hooking points were also added within the kernel, for third-party tools written in specialized languages permitting the analysis, synthesis, verification and viewing of models. It is possible to execute unmodified SystemC models in cooperation and in parallel with our models in the same binary file.

Compared to SystemC, the power of ESys.NET relies mainly on its direct connection to a meta-data oriented standard intermediate format, on the multi-language features of the .NET, and on the expressiveness, the reflectivity and the attribute programming model provided by the C#. These advantages allow us to consider very interesting perspectives for the future development of ESys.NET.

Our future work will have as prime objective the exploration of refinement under its many facets.

Firstly, the standard intermediate format will permit the creation of new refinement tools that can be guided in better ways by the use of meta-data that annotate and extend the manipulated models. Moreover, in order to bridge the gap between our environment and other existing tools (and environments) for hardware synthesis and model analysis (ex. CoCentric), we will focus on the automatic translation of C# specification to SystemC and/or VHDL models. We have recently started working on a new approach that will extend the Esys.NET environment with hw/sw partitioning features. The implementation is facilitated by the use of attributes (for partitioning directives) and the reflective capacities of .NET (for in-line checking of the respect of constraints and even system hw/sw configurability).

Concerning the issues of simulation, we intend to prove the multi-language and distributed capabilities of .NET by exploiting them in the study of software application and operating systems executed at different abstraction levels.

In the mid term, our team will address the problems surrounding the verification. When we designed the simulation core of ESys.NET, we had in mind the future needs required by verification and analysis tools; for instance, many hooking points within the simulation kernel have been provided for external tools use. Also, several APIs enable the introspection of the system's status at any given moment during a simulation. Due to the limited space, examples and details of the environment were omitted, for more information refer to http://www.esys-net.org.