# MILAN: A Model Based Integrated Simulation Framework for Design of Embedded Systems*

V. K. Prasanna
Department of EE-Systems
University of Southern California
Los Angeles, California
prasanna@usc.edu

C. Raghavendra
Department of EE-Systems
University of Southern California
Los Angeles, California
raghu@usc.edu

Akos Ledeczi
Institute of Software Integrated Systems
Vanderbilt University
Nashville, Tenesse
akos.ledeczi@vanderbilt.edu

## ABSTRACT

We present MILAN[1], a model based extensible environment that facilitates rapid evaluation of various performance metrics of a large class of embedded systems, at multiple levels of granularity, by seamlessly integrating different widely used simulators into a unified environment. MILAN promotes a high-level representation paradigm for specification of embedded systems, an integrated model-based approach towards system design, and a unified framework for simulation and design evaluation. In this paper, we provide an overview of MILAN, discuss Model Integrated Computing as a suitable technology for development, and illustrate the high-level modeling concepts being developed in the MILAN project, for embedded systems design and evaluation.

## 1. INTRODUCTION

High performance, multi-functionality, reduced power dissipation and low cost are critical requirements that need to addressed for design of embedded systems. System-on-Chip(SOC) architectures combine high performance with low power consumption and are thus suitable candidates for development of these systems. A SoC architecture is defined as a complex IC that integrates multiple functional units(RISC processors, configurable logic, customized functional units, on-chip memories etc.) of a complete end product onto a single chip. However, the heterogeneous nature of the components makes development of applications onto these architectures a challenging task. On one hand it provides a designer with increased flexibility and control for mapping applications, but on the other hand the designer is required to have an understanding of the behavior and functionality of the various components.

The state-of-the-art design technologies adopt an ad hoc approach towards system optimization and implementation [3]. Programming models and design tools for each hardware component are utilized independently to map an application task onto that resource and integration is performed much later in the design cycle. Lack of system level simulation and design tools increase the complexity of the design and development process. The performance analysis is done manually using component specific simulators [9, 12]. This is a tedious process since each simulator requires the designer to use a different interface and execution platform. Moreover, separate simulators are utilized for estimating the multiple performance and power metrics. This approach results in sub optimal solutions as global optimizations are difficult to make in such an environment. Moreover it is a time consuming process and is not capable of traversing the entire design space in the available time. Thus, selections are often made in an ad hoc, intuitive manner.

This paper introduces the **M**odel based **I**ntegrated simu**LA**tio**N** (MILAN) framework to facilitate embedded system design and optimization. MILAN is a collaborative project between the University of Southern California (USC) and the Vanderbilt University (VU).

The MILAN framework promotes system-level integrated techniques for design and development of embedded system on SoC architectures. System level programming and computation models [7, 2] are exploited for mapping applications onto resources in an optimal manner. The optimal solution is identified through a systematic exploration of design alternatives based on simulation and performance estimation. This is made possible by development of a unified simulation framework that seamlessly integrates different widely used simulators and facilitates end-to-end system modeling and simulation through a single, easy-to-use graphical interface. In addition to a unified user interface, MILAN permits a user to perform multi-granular simulation which provides the (simulation) time versus accuracy tradeoffs through integration of multilevel simulators. Moreover, the framework

*This work is supported by the DARPA Power Aware Computing and Communication Program under contract F33615-C-00-1633 monitored by Wright Patterson Air Force Base.

[1] http://milan.usc.edu

provides a feedback mechanism where results from a simulation can be utilized for updation of model parameters for better designs. In addition to performance and cost, this framework also incorporates power as an important design metric. Power optimization and estimation is facilitated through development of power models which are utilized for integration of existing component specific power simulators.

Presently, the focus of the project is on model development and simulator integration for application development and not on architecture synthesis. While an ideal environment would be a system-wide simulator that handles all the components: functional, storage, I/O, peripherals (A/D converters) among others, the mode of operation (analog, digital, RF) of components makes it difficult to realize such an environment and presently we only consider on-chip components and only digital modes

The remaining paper is organized as follows. Section 2 describes some of the related projects. The overview of the MILAN architecture and outline of its usage and scope is presented in Section 3. Sections 4, 5, and 6 provide technical details on system representation, design space exploration and simulator integration using MILAN. The present status of the project is discussed in Section 7, followed by our conclusive remarks in Section 8.

## 2. RELATED WORK

Several research efforts have focused on the study of system-level tradeoffs and performance optimization techniques. Notably, various hardware/software codesign groups have developed several formal techniques for the design of embedded system. The Ptolemy project [8] focuses on modeling, simulation, and design of heterogeneous embedded systems through a integrated framework of several computational models. These models represent the computational behavior of the components of a candidate system. MILAN utilizes the MIC [11] technology which is at a higher abstraction level than the Ptolemy models. While Ptolemy achieves coherency by imposing interaction constraints among components of a system by means of computational models, MIC provides flexibility to the developer to define own representation/interaction structure best suited to the system before modeling. As the MILAN project plans to integrate various heterogeneous simulators and evaluation models, the flexibility offered by the MIC technique facilitates development of scalable modeling paradigms satisfying diverse system requirements.

The POLIS project [1] provides a hardware software co-synthesis tool for design and synthesis of embedded micro-controllers. Chinook [4] is a hardware-software co-synthesis CAD tool for control dominated, reactive embedded systems, with emphasis on IP integration. However, these tools do not provide a unified environment where various simulators and models can be integrated to behave as a single system. MILAN hides the complexity of simulator and model integration and facilitates an application design and optimization environment. While POLIS and Chinook are tools for embedded system synthesis, MILAN provides a framework for application development on programmable embedded architectures. The use of MIC technique, makes MILAN scalable and easily modifiable to accommodate future architectures while providing the same interface for application development.

Simulink [10] is a commercial tool developed on top of Matlab for system modeling, simulation and analysis. This tool facilitates functional simulation, with a rich graphical user interface. MILAN, along with a graphical interface for systems design, integrates various other simulators for different performance metrics including Matlab. MILAN also hides the low level architectural details and exposes the parameters and performance attributes needed for application optimization.

## 3. OVERVIEW OF THE MILAN ARCHITECTURE

MILAN is a model-based, extensible simulation framework that facilitates rapid evaluation of different performance metrics, such as power, latency, and throughput, at multiple levels of granularity for a large set of embedded systems by seamlessly integrating different, widely-used simulators into a unified environment. MILAN is built using Model Integrated Computing (MIC) technology developed at Vanderbilt University previously [11].

MIC allows designers to create domain-specific models of systems, validate these models, and perform various computational transformations on the models. These formalized models capture various aspects of the system's desired behavior. Model interpreters are used to translate these models for use in the system's execution environment, either as stand-alone applications, as data sources for third-party off-the-shelf software packages, or in conjunction with code libraries and some form of middleware (e.g. CORBA, POSIX, etc.). When changes in the overall system require new application programs, the models are updated to reflect these changes and the applications are regenerated automatically from the models.

The Generic Modeling Environment is a configurable graphical tool suite supporting MIC [5]. The configuration of the environment to support a domain-specific modeling formalism is done in a formal manner through the use of metamodels. The metamodeling language is the UML class diagram notation. Well-formedness rules that are also part of the metamodels are specified using the Object Constraint Language (OCL). These constraints, along with the syntactical rules of the domain language, are enforced by the automatically generated target environment. MILAN exploits the MIC technology to present an environment tailored for embedded system design, evaluation, and optimization.

Figure 1 shows the architecture of the MILAN framework. At the top is the Generic Modeling Environment configured to support the modeling language developed specifically for MILAN. There are three kinds of representation models in MILAN: resource models, application models and explicit constraints. The resource models describe the available hardware components and their interconnectivity in a hierarchical block diagram-like notation. The application models are based on a hierarchical signal flow representation with important extensions. Most notably, the modeling language allows for the specification of explicit design or implementation alternatives of any component. This enables

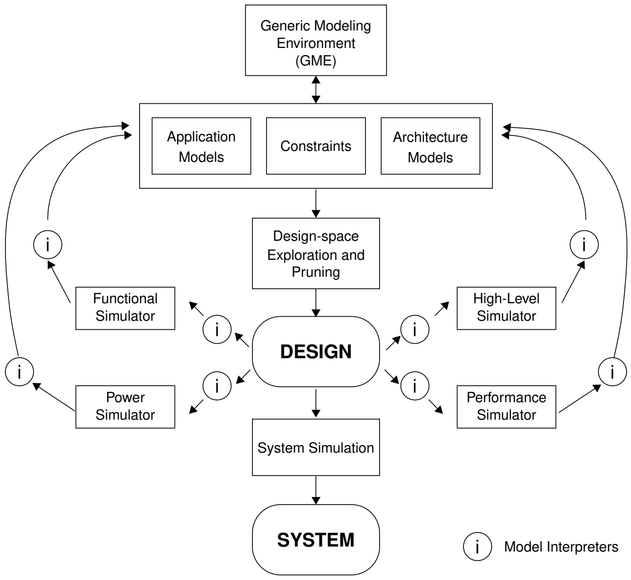


**Figure 1: MILAN Overview**

modeling of the entire design space of the application as opposed to a point solution. To manage this design space, application requirements, resource constraints and other specifications are captured as explicit constraints in the models.

The Design-Space Exploration and Pruning tool takes the potentially very large design space and applies the constraints using a symbolic constraint satisfaction technique to find the set of solutions that satisfy all the constraints. The modeling methodology and the design-space exploration technique are described in detail in subsequent sections.

The goal of design space exploration is to identify a small number of valid candidate designs. To find the balance between an under-constrained and an over-constrained model is a highly iterative, human-in-the-loop process. One of the design goals of the modeling environment and the design-space exploration tools is to support this activity.

The next step in the design process is to utilize the integrated simulators to simulate the candidate designs one-by-one. Each supported simulator has a corresponding model translator (or model interpreter, to use MIC terminology) that configures the simulation from the system models.

MILAN supports different classes of simulators. Functional simulators, such as Matlab or SystemC, verify the functionality of the application. The integrated high-level simulator provides a rapid, reasonably accurate estimate of different performance criteria of the system. This component of MILAN is described in a subsequent section. Lower-level power and performance simulators, such as SimpleScalar or SimplePower, are also supported. While they can be very accurate, their slow speed may prevent the simulation of all possible design choices. MILAN allows multi-level simulation, which exploits the trade-off between accuracy and speed. MILAN also supports the simulation of user selectable parts of the system.

One of the major challenges of an integrated simulation framework like MILAN is how to interpret the results of dissimilar simulators. In our architecture, model interpreters specific to each individual simulator take results and feed them back to the models. For example, results from a SimpleScalar simulation of a component can be stored in the models in the form of performance attributes that the high-level simulator can utilize in evaluating the performance of the whole system. Note that the interpretation of the results can be a human-in-the-loop process. For example, we do not plan any automatic model modification based on a functional simulation in Matlab.

Once the final design has been selected, the target application can be automatically synthesized. This step is fairly similar to driving the simulators. Instead of the semantics of the target simulator, the semantics of the runtime system have to be observed by the synthesizer.

Along with the simulators, MILAN integrates various evaluation and estimation models into the framework. These models are different from the representation models described in Section 4. These are primarily mathematical models that facilitate rapid evaluation of various performance attributes, inter-simulator communication, rapid design space exploration, and feedback mechanism. Some of the models are described below.

The *MILAN Performance Model* is based on a generic heterogeneous SoC architecture and characterizes it based on some key parameters. This heterogeneous system consists of a set of processing components and memory module connected to an interconnect. The set of processing components consists of one or many general-purpose processor core, application specific accelerators, and configurable components. Each component is characterized with critical parameters such as voltage, frequency, start-up cost, etc. The performance model is utilized to evaluate system design for a quick estimation of critical performance parameters without the use of any cycle accurate simulation. This quick estimation is enabled by use of functions or heuristics evaluating various performance attributes based on architecture parameters. For example, rapid prediction of performance and power consumption of various cache configuration is possible through a fast estimation equation generated using linear approximation [6].

Communication between simulators integrated into MILAN poses a major challenge. Typically simulators are not designed to interact with each other. The *MILAN Communication Model* advocates a common data-exchange layer between the simulators. This layer is responsible for translation of outputs from the source simulator to an orthogonal format, which is in turn translated to simulator specific format for the target simulator. An orthogonal format also makes integration of a new simulator an easy task. This communication model is based on the resource model and computation models described in Section 4.

The design space exploration and feedback models are described in later sections. All the models adhere to a common system representation scheme for input data. This representation facilitates a uniform input format for various simulators and tools integrated into the system.

# 4. SYSTEM REPRESENTATION

As discussed earlier, a representation method has been selected that partitions the system into three distinct classes of models: application models, resource models and constraints. Application models describe the task to be performed while the resource models describe the physical hardware available. Constraints specify requirements. A mapping between components of the application and available resource models is used to capture the space of possible design choices limited by the design constraints. The three classes of models are described in more detail below.

## 4.1 Resource Representation

Resource models represent the hardware components of the system. Hardware components typically consist of general-purpose processor cores, application specific accelerators, interconnect, memory, and configurable logic among others. Current advances in the hardware design area have enabled design of hardware with various flexible parameters (e.g. variable frequency, voltage, memory size). These parameters are critical for system optimizations and need to be identified and their effect should be evaluated very early in the design cycle.

The MILAN resource model is a hierarchical classification of hardware components related to embedded systems design. A few generic high-level components constitute the top level of the hierarchy. They are processing element, storage element, interconnect, sensor, and actuator. All possible hardware components for embedded system design fall under one of these categories. Each component is associated with a set of parameters identified after an extensive survey of the available hardware components.

To facilitate a logical structure for the resource model the modeling follows a design technique similar to the object oriented design. The model consists of specialized components, which inherit the parameters of the base component and are further specialized with additional capabilities. For example, an FPGA and a RISC core are both processing elements, but FPGA has lot of different features, namely the flexibility to be configured to suite the task under consideration. So going down the resource model hierarchy the FPGA branches out to fall under a separate category, the category of configurable processing elements. While frequency and voltage are some of the common parameters for the processing element as such, configuration cost (time and energy cost to reconfigure a configurable element) and hardware area usage are some of the parameters of interest for the configurable elements. Figure 2 shows the resource meta-model and the model of a traditional processor with ALU and cache modeled for MILAN. This model represents a resource that can be simulated using cycle accurate simulators like Simple Scalar or Simple Power.

Another interesting characteristic to be captured by the resource model is the identification of parameters, which can be varied to optimize the system. For example, voltage is constant parameter for many components, whereas for devices like SA-1100 it can be varied. These types of variable parameters define the design space for system design. The

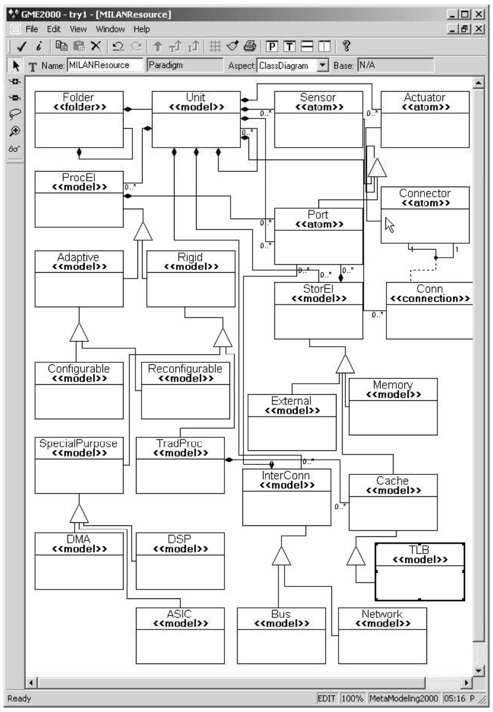


**Figure 2: Resource Modeling Paradigm**

more the number of parameters, the bigger is the design space. The choice of parameters depends on several factors. Some of them are:

- Desired detail of simulation
- Capability of the resource
- Parameters explicitly manipulated by the application
- Availability of simulators with desired capabilities
- Range of design choices to explore.

## 4.2 Application Representation

Application models are used to represent the processing algorithm to be implemented by the resulting system. Many different types of application representations may be used to represent embedded systems. We have chosen to initially focus on an enhanced hierarchical signal flow notation. These models compartmentalize a system into distinct components with well-defined interfaces. The "ports" that form these interfaces allow data to be exchanged between components. The signal flow defines the order of processing for an application. Each component in the signal flow graph receives data from other components, performs some transformation on the data, and then outputs new data to other system components. This modeling formalism is widely used in modeling embedded signal/image processing systems.

The following enhancements are included:

- Both asynchronous and synchronous dataflow semantics are supported.
- Signals are strongly typed. A separate sublanguage allows the precise specification of data types that are

then associated with ports. The environment enforces the type consistency of connections.

- Component level functionality implemented in hardware (i.e. configurable logic) can be expressed in a sublanguage that provides concepts that support SystemC and VHDL simulations of the components.

- The modeling paradigm allows the specification of explicit implementation alternatives at any level of the hierarchy. For example, a filter may be implemented in the time or the spectral domain, it may be implemented on a DSP chip in assembly language, a RISC processor in C, an FPGA or an ASIC, etc. These alternatives, each with different performance characteristics and resource constraints, can be captured in the models. Alternatives allow the environment to support the modeling of the design-space of the application, as opposed to a single-point solution.

- The environment supports multi-granular simulations by allowing the user to specify implementation scripts at any level in the hierarchy. Implementation scripts can be in C, Java, Matlab, SystemC or VHDL. Specifying these is mandatory at the leaf level; this is the information that is utilized during system synthesis. However, the user may choose to provide a C implementation of a high-level component directly, in order, for example, to drive the detailed simulation of the next component in a pipeline. This provides fine control over simulation granularity.

### 4.3 Constraint Representation

Design constraints capture system requirements such as timing, performance, power, cost, etc. Moreover, resource constraints and other information also need to be specified as part of the models. In MILAN, the Object Constraint Language (OCL) is used to specify constraints in a formal manner. Even though they utilize the same language, these constraints are different from the ones that are part of the MILAN metamodels which specify the static semantics of the MILAN modeling language. On the other hand, design constraints specify information about the target system.

## 5. DESIGN SPACE EXPLORATION

The approach we have chosen for rapid exploration of large design spaces relies on a symbolic representation based on Ordered Binary Decision Diagrams (OBDD). In this symbolic representation, a set (or space) is represented mathematically by its characteristic function. By introducing a binary encoding scheme over the members of the set, the set can be represented as a Boolean function. OBDD-s represents Boolean functions as directed acyclic graphs in a memory-efficient format. Fast graph algorithms implement operations over the Boolean function. Constraints express complex relationships and bounds over composite properties of elements. Given the binary encoding of the elements, it is feasible, albeit non-trivial, to represent constraints symbolically as a Boolean function over the encoding of the elements. Constraint application is a logical conjunction of the two functions. The resultant Boolean function represents the pruned design space. The principal advantage of this approach is that constraint satisfaction is accomplished without the enumeration of the entire space. Obviating the need for enumeration makes the approach highly scalable, and particularly attractive to representing extremely large design spaces. The scalability can be further improved by enabling hierarchical constraint satisfaction. In spite of the excellent scalability, the approach remains susceptible to a well-know problem, the "exponential blow-up" of OBDD-s. While further investigation is desired to alleviate this problem, initial results from phase-transition studies indicate that an order parameter may be found and used to identify and prevent the onset of an exponential blow-up.

## 6. SIMULATOR INTEGRATION

MILAN will enable *hierarchical simulation*, i.e. simulation of a set of task-to-resource mappings at different levels of granularity, from high-level coarse-grained performance evaluation to low-level cycle- or timing-accurate simulators. Employing high-level performance estimation tools that can (possibly) compromise the accuracy of results is beneficial since they enable a HW/SW trade off analysis earlier in the design cycle, prior to detailed simulations that are typically time-consuming.

There are two aspects to the creation of a unified environment for multigranular, system-wide simulation - *vertical* and *horizontal* simulation.

Vertical simulation refers to multigranularity, and primarily deals with the issues in interpreting the results of low-level, component-specific simulations in a global context. The assumption here is that all the simulations are run independently of each other, i.e. there is no data exchange or any other communication between simulators at any granularity level. Component-specific low-level simulators might not be capable of representing system-wide effects that have an effect on overall system performance (like communication delays between components). Therefore, it is necessary to have a suitable methodology for interpreting results of the low-level "isolated" simulations in a global context.

Horizontal simulation concerns itself with the challenges of making simulators interact with each other during concurrent execution. This aspect is not an immediate focus of the MILAN project, primarily because none of the existing widely-used simulators (to the authors' knowledge) lend themselves to easy modification that will enable them to trap selected events and efficiently interact with similar simulators in real time. A possible approach towards solving this problem is the definition of a standard API for simulators to synchronize on events, and either modifying existing simulators or requiring new simulators to confirm to the standard.

Ongoing research on vertical simulation in the MILAN project is focused on modeling computation costs on a specific component, and communication costs within components, at a high-level. This is because the high-level performance estimation tool is possibly the only component in the framework that can quantify the global performance effects mentioned above. The initial inputs for the computation cost can be provided directly by the user from vendor-supplied information, or obtained from offline low-level simulations, or from experiments on real architectures. The communication cost between consecutive stages depends upon the size of the data being passed, the rate at which data is made available by the previous stage, the rate at which it is processed by the receiving stage, the bandwidth of the interconnect, data reorganizations if any, etc. Average access latency can be analytically modeled or experimentally determined for various communication mechanisms like message passing, shared memory, etc.

Once high-level performance estimates are obtained, functional simulation is carried out through tools like Matlab. Functional simulation verifies the correctness of the application, and can also be used to obtain the actual data sets that are available at the inputs and outputs of each task-to-resource association. This information about data sets, coupled with the communication and computation characteristics of the system, makes it possible to suitably modify the results of low-level simulation before updating the model database, thereby enabling vertical simulation.

## 7. STATUS

MILAN is a collaborative project between the University of Southern California (USC) and Vanderbilt University (VU). The MILAN modeling paradigm leverages previous ISIS [11, 5, 13] and USC work [2, 7] in the DARPA Adaptive Computing Systems (ACS) program. The modeling concepts can be divided into mathematical models that represent the system in terms of parameters and functions, and constructive models that concern themselves with system components and their interconnections. The high-level mathematical modeling paradigm will be based on an extension of the hybrid system architecture models developed at USC [2]. The constructive modeling paradigm developed at ISIS will be based on the extensive experience of the ISIS team in modeling a wide variety of embedded systems using MIC [11]. Currently we are integrating SimpleScalar, Matlab and SystemC in the MILAN framework. A high-level, fast and reasonably accurate power/performance evaluation tool is also under development. A toolkit for supporting the integration of simulators by third parties are also planned.

## 8. CONCLUDING REMARKS

The MILAN framework will deliver a much-needed evaluation and integrated simulation platform for embedded systems. Model Integrated Computing (MIC) will allow the system to *evolve* with the ever-changing simulation requirements of embedded computing. Currently we are integrating SimplePower [12], SimpleScalar [9], Matlab, and SystemC into the framework. We have developed the preliminary version of the resource and application representations. These representations are closely related to power and performance models of available architectures. There are a potentially large number of simulators (most of them developed in an *ad hoc* manner) which the simulator integration framework will be required to support. Therefore, along with the framework we are also developing generic guidelines for simulator development and integration along with an *extensibility toolkit* to aid in extending MILAN. We expect a preliminary version of the framework to be available for public use by the end of the year.

## 9. ADDITIONAL AUTHORS

A. Bakshi, V. Mathur, S Mohanty, M Singh, email: {`abakshi, vaibhav, smohanty, mitali`}`@usc.edu`) and A. Agrawal, J. Davis, B. Eames, S. Neema, G. Nordstrom, email: ({`aditya.agarwal, james.r.davus, brandon.eames, sandeep.k.neema`}`@vanderbilt.edu`).

## 10. REFERENCES

[1] F. Balarin et al., "Hardware-Software Co-Design of Embedded Systems: The POLIS Approach," Kluwer Academic Publisher, Massachusetts, USA, 1997.

[2] K. Bondalapati and V. K. Prasanna, "Mapping Loops onto Reconfigurable Architectures," International Workshop on Field Programmable Logic and Applications, Talinn, Aug-Sep 1998.

[3] H. Chang, et al., "Surviving the SOC Revolution - A Guide to Platform-Based Design," Kluwer Academic Publishers, 1999.

[4] P. Chou, et al., "IPCHINOOK: An Integrated IP-based Design Framework for Distributed Embedded Systems," DAC, 1999.

[5] Generic Modeling Environment, http://www.isis.vanderbilt.edu/projects/gme/default.html.

[6] T. D. Givargis et al., "Fast Cache and Bus Estimation for Parameterized System-on-a-Chip Design", Design, Automation and Test in Europe 2000.

[7] The MAARC Project, http://maarc.usc.edu.

[8] Ptolemy Project, http://ptolemy.eecs.berkeley.edu.

[9] SimpleScalar Tool Set, http://www.cs.wisc.edu/ mscalar/simplescalar.html.

[10] Simulink 4, http://www.mathworks.com/products/simulink/.

[11] J. Sztipanovits and G. Karsai, "Model-Integrated Computing," IEEE Computer, April, 1997.

[12] N. Vijaykrishnan, et al., "The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool," Design Automation Conference, May 2000.

[13] Bapty T., et al., "Model-Integrated Tools for the Design of Dynamically Reconfigurable Systems", ISIS Technical Report/Vanderbilt University, 2000.