RASSP Virtual Prototyping Appnote

Abstract

Efficient design of complex systems requires modeling at multiple abstraction levels. This RASSP application note describes virtual-prototyping and how to use it within a rapid-prototyping environment. An example abstract-level virtual-prototype is examined. The prototype is produced as a natural extension of the token-based performance model. It represents the next logical step in the top-down design process and exemplifies techniques for seamless transition of design information in the flow-down process. The described rapid-prototyping methods were developed to permit earlier validation and more rapid product evolution. The virtual-prototyping techniques were developed on the RASSP project and have demonstrated simulation speed improvements orders of magnitude over earlier methods.

Purpose

This application note should be read by system architects, software designers who are responsible for validating software partitions among multiple computer nodes, and hardware designers who are responsible for designing system network configurations and components. The basic concepts can be applied to other systems and other levels of abstraction.

RASSP application notes augment course modules and case studies about digital system design. The material is applicable to the design of complex systems such as digital signal processing (DSP) systems and other multiprocessor systems. The application notes serve to document the design methods that were developed on the RASSP program. This application note describes the purposes and methods for *virtual-prototyping* .

Roadmap

 Dennis Basara

RASSP Virtual Prototyping Application Note

1.0 Executive Summary

Efficient design of complex systems requires modeling at multiple abstraction levels. This RASSP application note describes virtual-prototyping and how to use it within a rapid-prototyping environment. An example abstract-level virtual-prototype is examined. The prototype is produced as a natural extension of the token-based performance model. It represents the next logical step in the top-down design process and exemplifies techniques for seamless transition of design information in the flow-down process. The described rapid-prototyping methods were developed to permit earlier validation and more rapid product evolution. The virtual-prototyping techniques were developed on the RASSP project and have demonstrated simulation speed improvements orders of magnitude over earlier methods.

This application note should be read by system architects, software designers who are responsible for validating software partitions among multiple computer nodes, and hardware designers who are responsible for designing system network configurations and components. The basic concepts can be applied to other systems and other levels of abstraction.

RASSP application notes augment course modules and case studies about digital system design. The material is applicable to the design of complex systems such as digital signal processing (DSP) systems and other multiprocessor systems. The application notes serve to document the design methods that were developed on the RASSP program. This application note describes the purposes and methods for *virtual-prototyping*

 Dennis Basara

RASSP Virtual Prototyping Application Note

2.0 Introduction to Virtual Prototyping

A prototype is any preliminary working example or model of a product, component, or system. It is often abstract or lacking in some details from the final version. Two main classes of prototypes are used in design processes: *physical prototypes* and *virtual prototypes* .

A *physical prototype* is a physical model of a product, component, or system. Traditionally, prototypes were physical models, as opposed to virtual-prototypes.

Examples of physical prototypes are: bread-boards, mock-ups, and brass-boards. Physical prototypes are characterized by fabrication times that typically require weeks-to-months and that typically require days-or-weeks to modify. Construction usually involves detailed design, lay out, board or integrated-circuit fabrication, ordering, and mounting via solder or wire-wrap. Additionally, programmable systems or parts require detailed target-software design of drivers and operating system, or programming PLAs, FPGAs, PROMS.

A *virtual prototype* is defined in the RASSP Taxonomy as a computer simulation model of a final product, component, or system. Unlike the other model type-names that distinguish models based on their characteristics, the term *virtual-prototype* does not refer to any particular model characteristic but rather it refers to the role of the model within a design process; specifically for:

- exploring design alternatives,
- demonstrating design concepts
- testing for requirements satisfaction/correctness

To be useful in a larger system design, a virtual-prototype model should define the interfaces of the component or system under design. As with any model, a test-bench should exist for regressive verification.

In contrast to a physical prototype, which requires detailed hardware and software design, a virtual prototype can be configured more quickly and cost-effectively, can be more abstract, and can be invoked earlier in the design process. Another distinction is that a virtual prototype, being a computer simulation, provides greater non-invasive observability of internal states than is normally practical from physical prototypes. Comparatively, virtual prototypes introduce some risk due to the possibility of modeling inaccuracy or incorrectness.

Virtual prototyping is the activity of configuring (constructing) and using (simulating) a computer software-based model of a product, system, or component to explore, test, demonstrate, and/or validate the design, its concept, and/or design features, alternatives, or choices. Specifically, the act of using the virtual-prototype model as if it where an example of the final (physical) product.

Virtual-prototyping is synergistic with rapid-prototyping because it shortens product evolution cycles from days or weeks down to minutes. The designer can determine the effect of design changes on the behavior of the final system as quickly as it takes to edit a file.

Approved for Public Release; Distribution Unlimited *Dennis Basara*

RASSP Virtual Prototyping Application Note

3.0 Example - SAR System Virtual Prototype

The Digital Signal Processing (DSP) subsystem of a Synthetic Aperture Radar (SAR) system was virtual-prototyped under the RASSP Benchmark-2 project. Details of the SAR project can be found in the SAR Case Study.The virtual-prototyping was conducted by Lockheed Martin's Advanced Technology Laboratories (ATL) in VHDL for relatively seamless transition to the down-stream detailed design processes.

The design risk assessment indicated the most significant challenges involved integrating and coordinating the various hardware and software elements into a system of multiple cooperating PE's. Because the hardware and firmware elements were selected from COTS products, they effectively became validated by default. Therefore, the design team considered the prototyping of the complete hardware-software multi-processor system to be more important than modeling the operation of an individual sub-section of the architecture.

Preliminary design and modeling produced a performance model of the integrated system. The performance model assisted in:

- partitioning the application software tasks,
- mapping the software tasks to processor elements,
- scheduling the tasks, and
- selecting the network architecture.

The performance model indicated that the selected partitioning/mapping/scheduling solution could satisfy the processing throughput requirements. This assumed that the selected solution would produce the correct numerical results. However, because the performance model does not actually perform the numerical computations, it could not verify the numerical correctness of the solution, nor could it show the data values processed at any point. Therefore the performance model could not serve as a total concept validation.

To further validate and develop the design solution, a more thorough prototype of the system was needed. Specifically, an abstract-behavioral model was needed to describe not only the timing and structure, but the functionality as well.

Because the performance model was carefully constructed in VHDL for extensibility, a new modeling effort was ***not*** needed to construct the prototype from scratch. Instead, the whole performance model was re-used by simply adding the missing functionality to it, as shown in figure 1. In this way, the performance model was extended to an abstract-behavioral model. It is abstract because it does not resolve the actual bit-representations of the values or structures. It is a behavioral model because it describes both functionality and timing aspects. See the VHDL Taxonomy document for more details on these concepts.

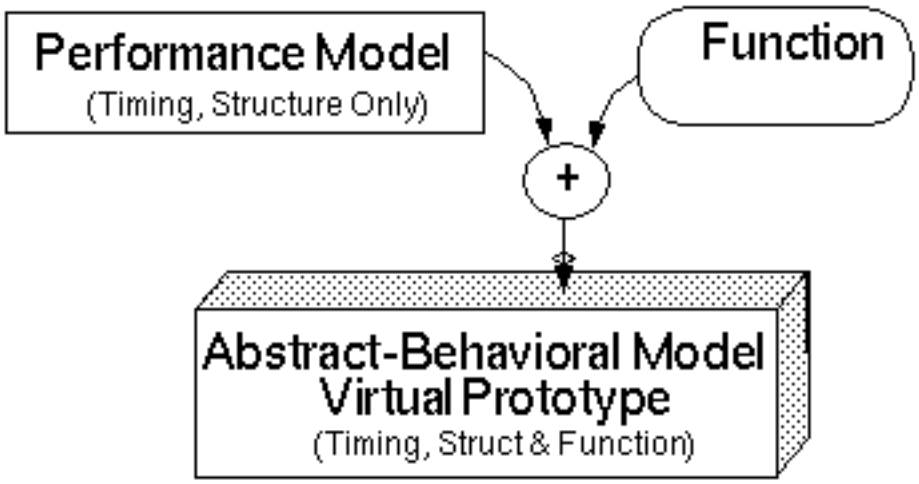

Figure 3 - 1: Extending performance model to abstract-behavior virtual-prototype

The new model served as a true prototype for the system. It responded in value and time as the designed target system would. It is a *virtual* -prototype, because it is not a physical construction, but exists as a software-based simulation.

Specifically, the compute subroutine, which consisted merely of a time-delay statement in the performance model, was augmented to include calls to the actual numerical subroutines, such as FFT and vector multiply. The token-definition was extended by adding one field to hold the data being transferred between processors.

An error in the assumed sequence of data arrival at a compute-node was discovered by the virtual prototype. This allowed the error to be isolated and quickly fixed prior to delivery of the hardware.

The abstract-behavioral model was later extended by incorporating a detailed register-transfer-level (RTL) model of the system I/O-board. This demonstrates the extensibility of this method, as well as the need to carefully consider the appropriate way to structure each abstraction level ahead of time.

When the physical hardware became available the software, which had been developed and tested abstractly on the virtual-prototype, was quickly ported and began running on the physical system. Because the application software had been pre-validated, some minor debugging issues were quickly isolated to a sporadic I/O device. The system was running correctly and meeting performance requirements within only about two weeks after porting began.

The Need for Abstraction

The VHDL virtual-prototype consumed 14 CPU-hours to execute 5-seconds of simulated runtime of a 6-processor version of the system. The simulated processor elements were Intel i860s which execute 40-million instructions per second. Because the virtual-prototype used abstract behavioral models of the processors, it does not explicitly model the individual instruction cycles. Considering the number of processors simulated, their instruction rate, and the duration of the simulation, the effective execution rate of the aggregate model was 23,810 instructions per second.

In contrast, a less abstract model of the processor, known as an instruction-set-architecture (ISA) model exhibited about 5.5 to 7.5 instructions-per-second on a Sparc-10 CPU. It is very clear that prototyping significant segments of the multi-processor system could not be practical with such a model. ISA models are more useful for understanding the behavior of software segments that dwell within a given PE. Table 3 - 1 compares the relative execution rates of the various model types.

In summary, rapid virtual prototyping of complex systems is made practical through appropriate use of

abstraction. The virtual prototype was found to accurately model the numerical results and time-related performance within a few percent of the eventual physically constructed system.

Simulation	Simulated Time	Host CPU Time	Equiv. Instructions/Sec
VHDL Performance Model of 24-PE System *	5.0-Sec	28-Minutes	2,857,000
VHDL Performance Model of 18-PE System **	5.0-Sec	18-Minutes	3,333,000
VHDL Abstract Behavior Model of 6-PE System **	5.0-Sec	14-Hours	23,810
ISA Model of One i860 PE Node *	5.0-mS	12-Hours	5

*i860-XR40-MHz, 40-MIPS
** ADSP-21060 Sharc 40-MHz, 40-MIPS

Table 3 - 1: Comparison of simulation efficiencies for types of models in the design process

---

Next Up Previous Contents
Next: Up: Appnotes Index **Previous:**2 Introduction to Virtual Prototyping