VHDL MODELING FOR SIGNAL PROCESSOR DEVELOPMENT

Cory Myers and Ray Dreiling
Lockheed Sanders, Inc.
P.O. Box 868
Nashua, NH 03061

ABSTRACT

This paper presents modeling approaches and experiences in the use of the VHSIC Hardware Description Language (VHDL) for the development of application–specific signal processors. Within our work on the ARPA/Tri–Service RASSP program we have developed and used VHDL modeling techniques for modeling the performance of signal processor systems and for the detailed design of signal processor systems. These approaches have been applied to modeling a large Infra–Red Search and Track system from a functional model, through performance modeling, through a full functional model, down to a detailed hardware implementation model.

1. INTRODUCTION

The ability of designers to rapidly develop and field application–specific signal processing is dependent on their ability to accurately model the systems that they wish to build. This modeling starts with modeling of the application to be developed and it continues through architectural analyses and into detailed design.

Accurate modeling of the system being developed is key to good selection of architecture and to rapid development of hardware. Good models of hardware speed development by reducing errors in design and by allowing simultaneous hardware/software development.

The Rapid Prototyping of Application Specific Signal Processors (RASSP) program is an ARPA/Tri-Service initiative to create a new process for the development of military signal processors [1]. The objective of RASSP is to dramatically improve the process by which complex digital systems, particularly embedded signal processors, are specified, designed, documented, manufactured, and supported. The program is focused on the development of a process for the conversion of an initial set of requirements to an optimum signal processor architecture design and embodiment while simultaneously enhancing the ability to perform seamless upgrades as requirements change. RASSP is also addressing the obsolescence problems created by the inconsistences between the life cycles of major systems and their supporting technologies.

RASSP's objective is not just to support prototype development, but to support full–scale production and life cycle system support. This means that RASSP must support full military system design, including Ada, quantity production, and support. It must provide a mechanism for capturing the complete behavior of the system in a form that can be inexpensively maintained and upgraded for twenty years or more without compromising performance or safety. RASSP must support large teams working on large projects, fulfilling their needs while capturing the benefits of a "skunk works" project development style. Accurate modeling, at all levels of abstraction, from the functional to the detailed hardware level, is key to achieving these goals.

2. OUR MODELING APPROACH

Our RASSP design methodology is derived from the traditional top down design paradigm with the incorporation of the Virtual Prototype concept [2]. The basis of this concept is to develop a complete description, in standard languages like VHDL and Ada, prior to fabrication. The design is checked out completely as a model prior to commitment to hardware. In this way design errors are caught when they are easy to fix and the system performance can be validated in simulation.

The choice of VHDL as the modeling language is important because VHDL provides:

Completeness: VHDL provides the mechanism for capturing the system behavior in a form that can be maintained and upgraded for twenty years or more; and

Portability: VHDL is an industry standard so models developed in VHDL can be ported to a wide range of simulation environments and can be maintained over the system's lifetime.

Our basic modeling approach is illustrated in Figure 1. The modeling begins with a functional description and proceeds through a series of refinements to produce detailed hardware and software. During this refinement process, as sequence of models are developed to model system function, system performance, system detailed behavior, and detailed system design.

2.1. Functional Modeling

The functional definition phase produces a data flow model that defines the systems behavior as a set of inter-

The Lockheed RASSP team is under contract to the Naval Research Laboratory, 4555 Overlook Ave., SW, Washington, DC 20375–5326. The Sponsoring Agency is: Advanced Research Projects Agency, Electronic System Technology Office, 3701 North Fairfax Drive, Arlington, VA 22203–1714. The Lockheed RASSP team consists of Lockheed Sanders, Inc., Motorola, Hughes, and ISX.

0-7803-2431-5/95 $4.00 © 1995 IEEE

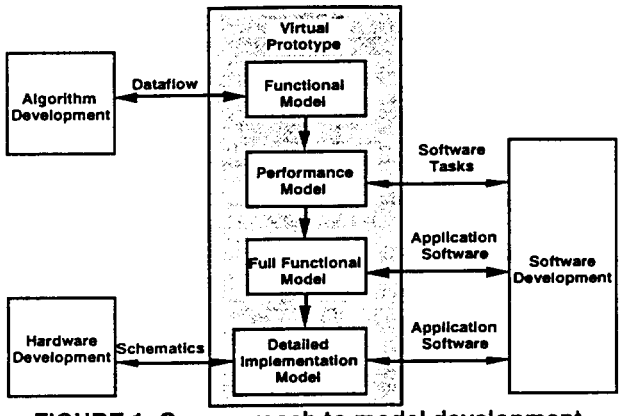

FIGURE 1. Our approach to model development starts with a functional model and refines it into a set of hardware and software descriptions.

connected sub-functions prior to hardware/software partitioning. These sub-function models are either used directly or translated for reuse in lower level definition phases. We have used VHDL modeling in the functional definition phase, particularly in the context of the design of a SAR image processor for purposes of benchmarking the RASSP Process [3]. The use of VHDL modeling for a functional specification of a signal processing algorithm is unusual. It has the following advantages:

Consistent Testing Environment: Our design process is based on VHDL modeling so the functional definition is captured in the same form that the hardware development will be captured in. This allows for later side by side comparison between the functional representation and the detailed hardware design within the same environment.

Path to Synthesis: For those portions of the functional specification which will be implemented in custom hardware, rather than in a programmable processor, the VHDL description provides a better starting point for the hardware synthesis problem.

2.2. Performance Modeling

The Performance Modeling phase examines candidate architectures for trade-offs in both hardware/software partitioning and architectural elements. Performance models incorporate abstract application software descriptions. We have used a set of VHDL performance modeling libraries developed for the Air Force [4]. This library defines five basic types of architectural elements, as follows:

Pipelines: These architectural elements model devices which implement first–in/first–out behavior. Their behavior is primarily characterized by a delay.

Memories: These architectural element model storage. Their behavior is characterized by their access time.

Bus Interface Units: These architectural elements model busses. Their behavior is characterized by a transmission rate.

I/O Devices: These architectural elements describe sources and sinks of data. They are characterized by their data rate.

Processors: These architectural elements describe programmable processors. They are characterized by their scheduler and by parameters of a set of resources.

An architecture is defined by connecting the architectural elements and mapping functional pieces to either dedicated hardware elements, which are modeled as pipelines, or to programmable processor elements. Functions mapped to programmable processors are modeled by specifying an abstract description of their resource and memory usage requirements. For example, an FFT implemented on a programmable processor may be characterized as requiring a certain number of floating point operations and a certain number of memory references. Characterization of processing elements and algorithms can be made data–dependent in a limited manner but the basic piece of data that flows between modeling elements, *a token*, consists of a marker that data has been produced, not the content of the data.

Architectural performance of a mapping functions to architectural elements is determined by running the performance model and recording statistics about processor utilization, bus utilization, processing latency, memory usage, throughput, etc. Performance modeling is used in conjunction with static analysis, as illustrated in Figure 2.

2.3. Full Functional Modeling

A Full Functional Model provides a model that is structurally correct and exhibits the functional and performance characteristics of the entities being modeled. At this level dedicated hardware elements are modeled by their behavior, not in a way that implies their implementation. For example, a dedicated filtering chip would be modeled in way that was correct bit–wise but did not imply the implementation structure.

At this level of modeling, we have been using Instruction Set Architecture (ISA) and Instruction Set Simulator (ISS) models. Our ISA modeling approach is to develop VHDL behavioral models of processor that can execute software and provide complete access to the internal registers of the processor [5]. Our ISS modeling approach is to integrate a commercial processor simulator into a VHDL environment. In either case, the processor model is combined with a Bus Interface Model (BIM), which models the detailed interaction of the processor at it connections, to make the full functional model. We have used this approach both to model individual chips, i.e., the i860, and to model single board computers.

Both the ISA and the ISS approaches allow the application software that is to run on the target hardware to be run in the simulation environment prior to physical hardware delivery. It is at this stage in the modeling process that detailed errors about the meaning of interfaces can be

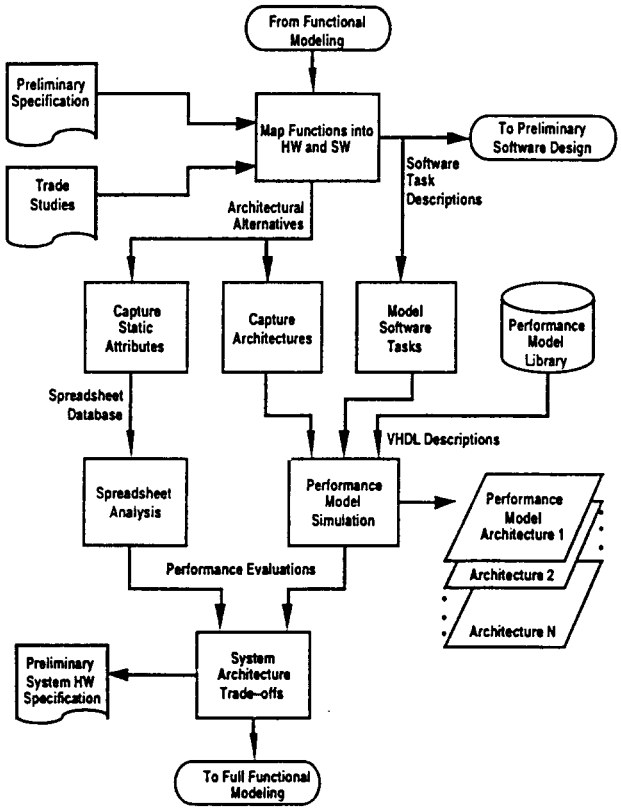

FIGURE 2. Development of architectural alternatives involves performance modeling of system.

identified and corrected. Additionally, this type of model gives the software much more accessibility to the state of hardware than is often the case when the software is run on the physical hardware.

2.4. Detailed Implementation Modeling

The Detailed Implementation Modeling provides models which are sufficient to determine the implementation of components. For example, at this level the exact structure of multipliers, adders, and registers would be obvious for a digital filter. This detailed model is derived from the abstract behavioral model by a combination of synthesis tools and manual design.

It is our approach to the modeling problem to only develop detailed implementation models for those components which we are developing. For components that are purchased, we use the abstract full functional model. Thus, a complete system simulation is a mixed level model.

3. RESULTS

Our experiences with the VHDL modeling approach includes development of a Infra–Red Search and Track (IRST) processor and a Synthetic Aperture Radar (SAR) processor [6,3]. Both processors have been developed from functional specification to hardware designs using

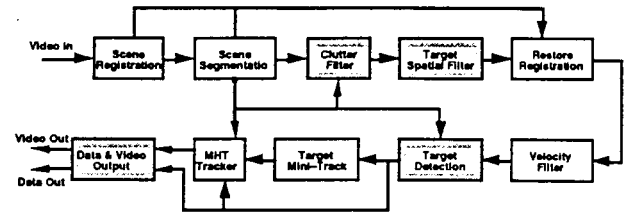

FIGURE 3. The IRST processing problem requires target detection and tracking. The shaded steps are required processing while the others are required only in heavy clutter environments.

the modeling approach. We will describe our work on the IRST processor in more detail here. Our work on the SAR processing problem will be completed by mid–1995.

3.1. The IRST Processor Problem

The IRST processing problem is illustrated in Figure 3. It takes input data from an infrared sensor at between 15 to 135 million sample per second depending on the sensor size. Input data is in fixed point format. The output rate is small, being just detected target message reports and graphic symbology. The image size is 2200 by 1800 pixels and processing takes place most in overlapping subimages of about 200 by 200 pixels.

The processing required for the IRST problem is a function of the sensor coverage, the scene revisit interval, the type of filtering algorithm, and the number of target movement hypotheses. Requirements can range from half a million to several hundred billion operations per second (BOPS). For our problem we require 5 BOPS.

3.2. Modeling for the IRST Problem

For the IRST problem we have developed a working system that is to be flown aboard a test aircraft. In terms of our modeling process we performed the following steps:

Functional Modeling: We developed a VHDL model of the IRST processing algorithm by translating an existing C code implementation.

Performance Modeling: We used the performance modeling approach to analyze three candidate architectures for the IRST processing problem. We examined the Mercury Raceway architecture, the Intel Paragon MP, and the ISI Embedded Variant and were able to determine that the most appropriate architecture for this problem was the Mercury Raceway (due to communications issues). The resulting architecture consisted of a Mercury Raceway with multiple MVC9 boards (16 i860 processors per board), dual video input cards, a video output card, and a system controller, as illustrated in Figure 4.

Full Functional Modeling: We developed a full functional model of the MCV9 board using an ISA model of the i860. We also developed behavioral models of the video input and output boards. Driver software was run on

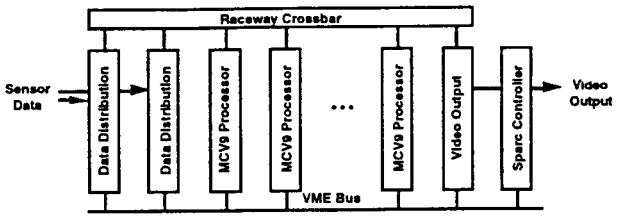

FIGURE 4. The IRST processor consists of video input, processing, video output, and a controller.

the i860 models and was used to control the video input and output models.

Detailed Implementation Modeling: We developed detailed implementation models for the custom components (FGPA programming) on the video input and output boards.

The total VHDL model consists of over 60,000 lines of code (LOC), including 4000 LOC for the i860 ISA model, 4000 LOC for Mercury's custom processor connection ASIC, 4500 LOC for Mercury's custom interconnect ASIC, and 5300 LOC for the VME logic. These models were developed by a team of eight people, geographically distributed among Lockheed Sanders, Hughes, and Motorola, over a period of nine months. The model is capable of processing 1500 simulated instructions per second for the ISA model using the Vantage VHDL simulator on a Sparc 10/40 workstation.

3.3. Lessons Learned in the IRST Problem

During this modeling process we have observed the following strengths in the modeling process:

Common Language: The use of VHDL performance modeling allowed the system engineers, the hardware engineers, and the software engineers all to interact on a common, executable, model of the processing problem. This eliminated many communications issues within the team.

Elimination of Hardware Errors: The development of a complete system model with the proper structure allowed the development team to catch and eliminate several hardware interface errors that would normally have been found after physical integration.

Early Software Debugging: The ability to run driver software on the behavioral model allowed system software debugging to start early. This process caught coding errors which otherwise would not have been caught until after physical integration. Additionally, the simulation environment has the potential to provide more access to what was happening in the hardware than is often found in the physical hardware.

Hardware/Software Codevelopment: The modeling approach allowed simultaneous development of hardware and software. Additionally, within the simulation environment the hardware and software engineers were able to easily negotiate the details of register formats and coding.

We have also observed some weaknesses in the modeling process, including:

Lack of Existing Models: Development of the performance models, the ISA model, and the behavioral models were time–consuming efforts. We often had to develop our own full functional models for COTS parts. Fortunately, much of what was developed here can be reused on other problems.

Lack of Software Debugging Tools: Our use of the ISA model for a processor did not allow the use of standard debugging tools. We are working to solve this problem by using an ISS model in our current SAR work.

Maturity of the Performance Modeling Library: The chosen library of performance models was not the most mature. The RASSP program is addressing this issue by funding commercialization of the performance modeling technology.

Large Simulation Resources Required: The VHDL simulations required a Sparc 10 workstation with 256 Mbyte of memory and 500 Mbyte of swap space. Simulation runs typically produced simulation files of 200 Mbytes. We are working to reduce these requirements by investigating alternative VHDL simulation technologies.

4. SUMMARY

In summary, we have presented a top–down approach to developing a complete VHDL model of a signal processing system. This approach has been used successfully for modeling the development of an IRST processor for our RASSP demonstration project.

REFERENCES

[1] J. Corley, V. Madisetti, and M. Richards, "Introduction to ARPA's Rapid Prototyping of Application Specific Signal Processors (RASSP) Program," in the Proceedings of ICASSP 1995.

[2] R. Dreiling, "Processes and Experiences in VHDL Top Down Design," in the Proceedings of the First Annual RASSP Conference, August 1994.

[3] B Zuerndorfer and G. A. Shaw, "SAR Processing for RASSP Application," in the Proceedings of the First Annual RASSP Conference, August 1994.

[4] Honeywell SRC, "Graphics Processor Definition VHDL Processor Model," AF Contract F33615–90–C–3800.

[5] S. Famorzadeh, T. Egolf, V. K. Madisetti, P. Kalutkiewicz, M. Falco, and R. Dreiling, "Rapid Prototyping of Digital Systems with COTS/ASIC Components," in the Proceedings of the First Annual RASSP Conference, August 1994.

[6] M. Vahey, "Image Signal Processor Demonstration," in the Proceedings of the First Annual RASSP Conference, August 1994.