RASSP Benchmark-1 and -2: A Preliminary Assessment*

A. H. Anderson, G. S. Downs, G. A. Shaw

Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02173-9108
shaw@ll.mit.edu

Abstract

These two benchmarks required the development of a virtual prototype and a hardware prototype, respectively, of a Synthetic Aperture Radar processor. The two RASSP Developers chose different approaches: one used COTS components on custom boards with a methodology emphasis on detailed VHDL prototyping and board design and one used COTS computer boards with a methodology emphasis on efficient VHDL modeling and automatic code generation. Both efforts are briefly described. A preliminary assessment of Benchmark-1, which has been completed, is offered with emphasis on the experience with VHDL modeling. Based on this assessment, some recommendations for improvement are made.

1: Introduction

The benchmarking component of the RASSP program is designed to evaluate the process, tool integration, and products developed, and to report the results to the Developers, sponsors and other interested parties[1]. In the course of the RASSP program, several application threads will be selected and design exercises defined for up to six benchmarks. MIT Lincoln Laboratory developed the specifications for Benchmark -1 and -2, and is observing and measuring the Developers' performance. The first application thread is a Synthetic Aperture Array (SAR) signal processor. Benchmark-1 requires that the developers perform trade-offs of several architectures and create a VHDL virtual prototype of their selected design. Construction of a physical (hardware) prototype of the SAR processor is required in Benchmark-2. Benchmark-1 has been completed and Benchmark-2 is scheduled for completion in August, 1995. This paper reports some of the conclusions from Benchmark-1, some early observations of Benchmark-2, and comments on the benchmark process.

* This work was sponsored by the Advanced Research Projects Agency, Electronic Systems Technology Office.

2: Benchmark Process

Lincoln Laboratory acts as the customer for the benchmarks. The deliverables include not only the virtual and hardware prototypes, but also extensive metric data on the design process, tool use and final products, as well as financial data. Besides participating in the periodic program reviews, Lincoln Laboratory personnel attend internal meetings, interview benchmark personnel, and collect copies of work in progress at defined milestones.

In addition to the usual documents prepared by a customer (a technical description, a list of deliverables and contractual details), Lincoln delivered a VHDL Executable Requirement at the beginning of Benchmark-1. During Benchmark-2, Lincoln will deliver to the Developers a hardware source/sink[1] with which to test the prototype.

3: SAR Processor

The processor specified for the SAR application thread is required to form images in real time on board an unmanned air vehicle, and to be compatible with the data rates and format of the Lincoln Laboratory Advanced Detection Technology Sensor (ADTS)[2]. The ADTS radar is a fully polarimetric air-to-ground SAR that operates in both "spotlight" and "stripmap'' mode; only the stripmap mode is employed for RASSP. The processor can be partitioned into seven functional blocks:

1. Preamble detection and extraction of radar and auxiliary data from the input data stream.

2. Video to baseband I/Q conversion.

3. Range processing.

4. Corner turn.

5. Azimuth processing.

6. Output data formatting.

7. Command processing, control and test.

At its maximum PRF of 556 Hz, the radar delivers 512 pulses with 4064 data samples for each of the four polarizations in 0.92 seconds. The processor must be able to form a 512-pulse image for each of *three* polarizations in real time with a latency not greater than three seconds. The processor can be set up to use anywhere from an 8 to a 48 tap FIR filter in it's video to baseband I/Q conversion. With a baseline algorithm the computational requirement with the 8 tap filter is about 1.1 Gflop/second, and with the 48 tap filter about 2.0 Gflop/second.

The processor must conform to a 10.5'' × 20.5'' × 17.5'' space with expansion space for twice the computational throughput, weigh less than 60 pounds, use less than 500 watts on average, and be designed to operate on-board a small aircraft. Test requirements were "best practice''.

4: Benchmark-1

The product of Benchmark-1 was described this way in the Benchmark-1 Technical Description[3]:

> Each RASSSP developer shall investigate at least two prototype processor designs, one minimizing cost to produce in prototype quantities, and the other minimizing processor power and weight. Both designs shall be developed to the point where realistic estimates of performance can be made. Use of VHDL performance modeling to substantiate the performance estimates is desired. Both designs must also be producible in unit quantities within the time and effort constraints established for Benchmark-2.
>
> One of the architectural concepts investigated at the performance model level shall be selected by the Developer for evolution to a virtual prototype as described in 1.1.1. Insofar as possible, subject to the six month duration and 5000 hour equivalent level of effort established for Benchmark-1, the virtual prototype shall emulate the critical behavior and timing of the selected design.

The Section 1.1.1 referenced above defined virtual prototype in the following manner[3]:

> As used here, a virtual prototype is an executable software model of an embedded processor which represents all of the important function and timing information of the processor with sufficient fidelity to insure that the processor will perform as intended when constructed in accordance with the architecture underlying the virtual prototype model. … One of the principal issues of interest in Benchmark-1 is the degree of fidelity and completeness that can be obtained for reasonable cost using existing virtual prototyping methodologies and models. The expectation is that IEEE-compliant VHDL simulation will be the vehicle for developing the virtual prototype. The level of detail and fidelity attained in the virtual prototype will be limited by the constraints of time (6 months) and level of effort (5000 person-hours) imposed for Benchmark-1. One of the responsibilities which the Developers have under the RASSP program is to establish cost-effective methodologies and tools for the creation and application of virtual prototypes to the development of embedded signal processing systems.

4.1: Executable Requirement

At the beginning of Benchmark-1 Lincoln Laboratory delivered to the Developers an Executable Requirement (ER)[4][5]. It was a VHDL behavioral model of the processor which implemented one possible algorithm. The ER modeled timing at the input and output data ports, and throughput latency. A VHDL test bench sourced commands and data from disk files, compared data outputs with supplied image files, wrote outputs to disk files, and checked latency of the processor. Data files from the ADTS radar and a synthetic data set were supplied, as were reference image files which had been generated by a C-language simulation of the SAR processor. The C simulator was not distributed. An image-pixel based limit on the difference between the reference processor images and the generated images was specified with an option given for the Developers to define and validate any other reasonable error metric. (Both Developers accepted the limit definition.) Two errors in the formatting of the input data, and one in both the C and VHDL simulators were not discovered and corrected until late in Benchmark-1.

4.2: Lockheed Sanders Process and Architecture

Lockheed Martin Sanders (Sanders) performed a standard-practice trade-off analysis using data from their experience and vendor data sheets to compare several different architectures[6]. A system based on the Sharp Elec-

tronics Corp. LH 9124/9130 DSP chip set was chosen based on low Life Cycle Cost, attributable to longer MTBF, and lower material cost. Six chip sets are used, one each for range and azimuth in all three processors (one for each polarization). The video-to-baseband conversion is done in the range processor. Input and output data processing is done with FPGAs, and FIFO memory devices buffer data between subsystems. SRAM memory devices with a custom controller are used for the corner turn function, and RACEway devices are used for data communication.

A block diagram of the system appears in Figure 1, where a 68040-based single board computer is used for

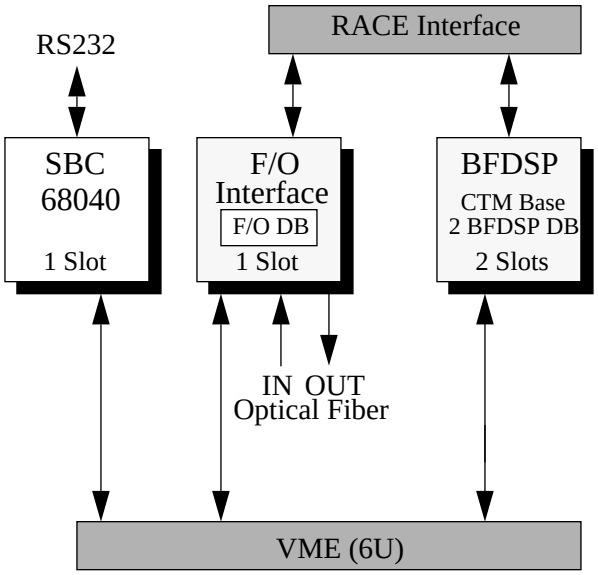

Figure 1. Block diagram of the Lockheed Sanders SAR procesor. DB = Daughter Board. □ = Custom Board. ■ = Standard Bus. □ = COTS.

the command processor. The block-floating digital signal processing (BFDSP) is distributed between 2 daughter boards on the BFDSP board, each containing 3 of the 6 Sharp processors. The corner turn memory is also on the BFDSP board. The fiber optic (F/O) board contains a COTS daughter board, while the mother board is a custom design.

Four of the five Xilinx FPGAs were designed in a bottom-up standard process, and the fifth was modeled in VHDL and synthesized. Sanders developed software to convert Xilinx XNF files to VHDL. Three printed circuit boards were designed: the F/O board, the corner turn mother board, and the BFDSP daughter board for range and azimuth processing.

Design of the Ada software for the command processor was done with Cadre TeamWork, while coding was done with standard practice methods.

The Sanders virtual prototype was written in VHDL with partitioning at the major subsystem level; for instance, there is one entity each for the range and azimuth processing for all three polarizations. The models are behavioral with timing at the interfaces. A model for the RACEway device was reused from the Lockheed Sanders RASSP demonstration project. In Benchmark-1, the command processor is modeled by a high-level VHDL model. A VHDL simulation to create one image of one polarization requires 11 days on a Sun SPARC-10 workstation[7]. An interface between a Microtec Instruction Set Simulator (ISS) of the command processor and Vantage VHDL has been implemented and demonstrated. In Benchmark-2 Sanders intends to build a virtual prototype with application code running on the ISS, and the remainder of the system in the VHDL simulation.

The Benchmark-1 development was done with version 0.1 of the Sanders RASSP Design Environment (RDE) which provided a GUI with launching capability for most tools. It also provided rudimentary logging of tool use and a personal-log data base tool, both of which were very useful for tracking tool use as well as development activity.

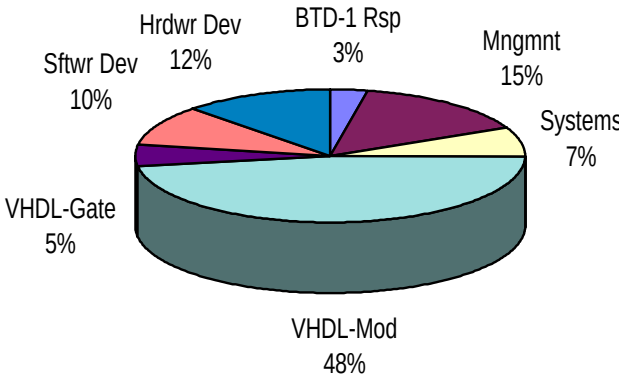

Figure 2. Effort distribution for the Lockheed Sanders Benchmark-1 (7812 man-hours as of 15 June 1995).

As shown in Figure 2, one half of the total benchmark effort at Sanders was devoted to VHDL modeling of the system, leading to the Virtual Prototype. The effort resulted in 13623 lines of code (LOC) as of 17 March (including executable lines, declarations, and specifications, but excluding comments). Approximately 2990 hours were expended as of that date on this effort[8], cor-

responding to a productivity of 36.5 LOC/person-day. Considerable follow-on effort has gone into further integration and testing of the virtual prototype, so the final value of this productivity measure will be lower.

Tools for evaluating the syntactical complexity of VHDL modules do not exist at this time, so these metrics are not available for the VHDL modules. However, such support is available for Ada code, and complexity metrics and LOC counts have been computed for the control software running on the control processor. A total of 3412 LOC (including executable lines and function definitions, but excluding comments) were written. Approximately 754 hours were expended as of 17 March on this effort[8], corresponding to a productivity of 36.2 LOC/person-day. (This value is somewhat preliminary since the Ada effort will continue in Benchmark-2.) Values of the McCabe cyclomatic complexity[9] metric, measured for 63 Ada modules, are summarized in Figure 3. The distribution of complexity values indicates most of the Ada modules are of acceptably low complexity, while 4 modules exhibit high complexity values. A high cyclomatic complexity implies these 4 modules are more likely than the low-complexity modules to contain defects. Not surprisingly, the highest complexity values are primarily associated with the largest modules, as illustrated in Figure 4. With only a few exceptions, there is a clear correlation of module complexity with size. Defect rates in these modules are being tracked in Benchmark-2.

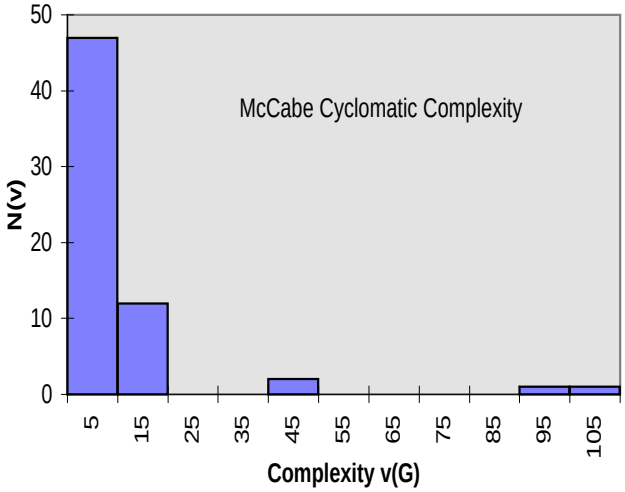

Figure 3. Distribution of the McCabe complexity metric for 63 Ada modules.

4.3: Martin Marietta Process and Architecture

Lockheed Martin Advanced Technology Laboratories (Martin) used CSIM, an in-house performance modeling tool, to determine the time line for several different architectures with different mappings of functions to HW and SW. An architecture with three Mercury Computer Inc. MCV6 boards (6U VME), each with four ADSP 2106X processor chips, with the I/Q FIR filter implemented in hardware, was chosen based on minimum size/weight for a COTS design and good expandability and upgradability.

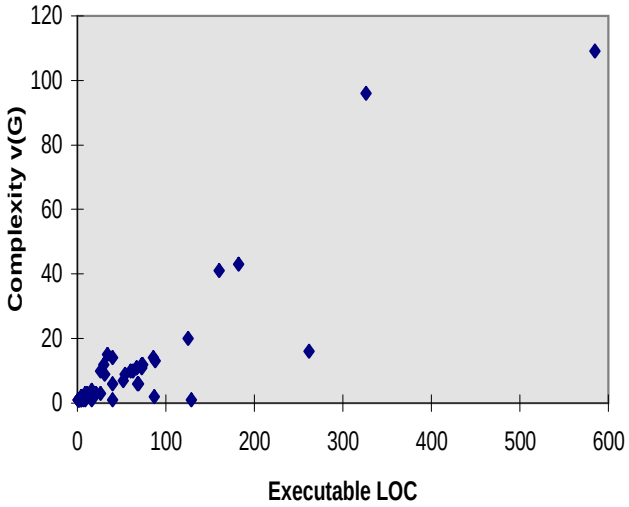

Figure 4. Correlation between McCabe complexity and module size (LOC).

The block diagram of the system is shown in Figure 5. A 68040-based SBC and an i860 processor (CE in Figure 5) on one of the Mercury boards does command and control processing. MATLAB models in four different precisions were written to determine the degree of precision required to meet the error specification. The processor uses single precision FP and the FIR filter 23 bit integer arithmetic. The input filter employs Plessy filter chips.

A VHDL performance model, similar in concept to the CSIM model, was written. It uses VHDL models of the signal processing functions and the RACEway, modeling the timing of data packets flowing in the system and the timing of the computations. The performance model comprises 1067 lines of VHDL code plus 509 comment lines, and two C programs of 438 total LOC. (The C programs create data files for describing software latency and data routing.) This model was expanded to a virtual prototype by adding data to the packets, a full behavioral model of the Data I/O board, and bit-true computation to the

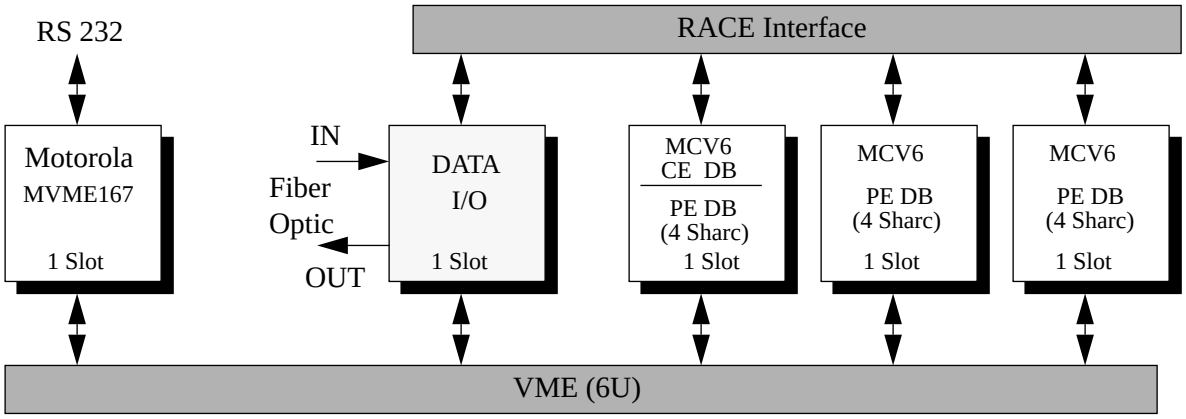

Figure 5. Block diagram of Martin Marietta SAR processor. DB = Daughter Board. ▢ = Custom Board. PE = Processing Element. ■ = Standard Bus. ▢ = COTS. CE = Control Element.

processor model. A modified test bench from the Executable Requirement was used. The command processor was not modeled. The performance model ran for 18 minutes on a Sun SPARC 10 computer while simulating the processing of three polarizations for five seconds of real time. The virtual prototype ran for 14 hours to produce three images of one polarization from five seconds of real-time data.

The virtual prototype comprised about 8000 LOC, of which 5200 are executable. This includes about 2400 reused LOC (1600 executable) from a math library and from the Executable Requirement. Including library development, 2027 hours were expended on this effort, yielding a productivity of 20.5 executable LOC/person-day.

The custom Data I/O mother board (Figure 5) has two daughter boards, one for the COTS optical cable transceiver, and one for the custom filter board. The I/O board was modeled in VHDL with behavioral models for COTS components and RTL models for the two AT&T ORCA FPGAs. The FPGAs were synthesized from the models. The VHDL I/O models comprised about 5700 lines of code of which 2800 were executable lines.

The RDD-100 tool was used to document requirements and assign functionality to HW and SW. The entire SAR system was modeled in PGM (Processing Graph Method) and a small data set simulated with PGSE (Processing Graph Support Environment)[10]. The command program was designed using the Schlaer-Mellor object-oriented approach and Cadre OOA/OOD tools.

Much of the Ada code was automatically generated (the command program), while C code for the control program (which runs on the i860 and controls the processing graph and data transfer) and the signal processing program were generated with standard methods. The interface between the command and control processors was hand coded, but in a style which will be used by later automatic methods.The programs were designed and tested so that testing could be done on both workstations and the target hardware. At the completion of Benchmark-1, 6980 lines of source code had been produced, of which 3726 were executable (43% of these were Ada code generated automatically, while the rest were hand-coded C and Ada), 2448 lines were comments, and 734 were reused[11]. At the end of Benchmark-1, about 1859 hours had accrued for this software effort[12], yielding a productivity of 16.0 executable LOC/person-day.

No integrated design environment or framework was used by the Martin Benchmark team.

The break-out of person-hours in Benchmark-1 appears in Figure 6. The VHDL and software (Ada, C)

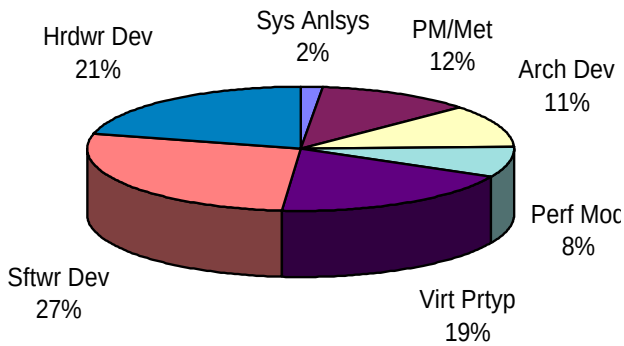

Figure 6. Effort distribution for the Martin Marietta Benchmark-1 (8636 man-hours as of 30 April 1995).

efforts are about equal in Benchmark-1. As in the Sanders effort, some hardware development was done in anticipation of the schedule for Benchmark-2.

5: Benchmark-2

The primary efforts for Sanders in Benchmark-2 are design validation and fabrication of the custom boards followed by system integration and checkout. In parallel, further work is being done on HW/SW cosimulation in an ISS/VHDL virtual prototype. Parts availability is not

expected to be a problem in meeting the BM 2 schedule, but the custom board schedule is ambitious.

The primary efforts for Martin are final software development, fabrication of two custom boards and integration and checkout with the Mercury Computer boards. Since availability of the Mercury boards with Sharc processors may present a schedule problem boards with i860 processors may be used.

6: Evaluation of RASSP Process

A comparison of the two Developers' methodology to standard practice can not be made until the completion of the hardware prototype. However, some general observations and evaluations can be made based on the completed Benchmark-1.

6.1: Methodology and Infrastructure

Since the RASSP methodology and tool integration were immature at the beginning of the benchmarks, a significant part of the effort of both developer teams was spent on methodology development and associated tool improvements. At Sanders, the effort was in VHDL modeling and cosimulation, while at Martin it was in performance modeling and automatic code generation. Although the benchmarking is intended to evaluate the current state of methods and tools, the work devoted to refining methodology and tools is clearly needed and is beneficial to the RASSP program.

6.2: VHDL Virtual Prototyping

VHDL modeling is a large part of both efforts. Availability of models for COTS components would have been of great benefit. Most important, perhaps, is the need for better understanding of the best ways to use modeling: what abstraction levels to use, how to efficiently evolve from one model to another with a greater level of detail, and how to design models so that scaled data sets can be employed for efficient debugging (as was done at Martin). Much VHDL methodology development must still be done by both developers.

The benefits of configuration control are not yet realized by all users of VHDL, and the lack of such control made tracking of VHDL development difficult. Because of modeling difficulties and the time pressures of the Benchmark-1 and -2 schedules, some of the detailed hardware design was done in parallel with the virtual prototype development. The full benefit of developing the virtual prototype, in terms of defect reduction and ease of integration, can not be evaluated until completion of the hardware prototypes.

6.3: Executable Requirement

The input data and images distributed with the ER were used by both developers, but the benefit was compromised by the errors described in Section 4.1. Martin used the processor model to generate intermediate data sets so that work on the azimuth processor was independent of work on the range processor. The ER test bench was used by both Developers, and both reused some of the code in the ER processor model.

7: Evaluation of Benchmarking

The principal goal of benchmarking is to measure the improvements made by RASSP compared to standard practice. Parametric Cost Estimators (PCE) are being used for this purpose[13][14]. To use PCE, good measurements of development and component costs are required. These can be obtained quite easily, but the development cost is biased by the effort described above to develop new methods and tools, and by the learning curve associated with the new technology.

Metric data is also collected for evaluation apart from PCE. Such data includes tool usage, measures of the quality of the product, degree of tool integration, etc.[1] The tool log in the Sanders RDE is a first step in automatic collection of tool data. Tools exist for measuring certain quality indicators of Ada and C code but not yet for VHDL[15].

Tracking of defects has been a challenge due to the short length of the benchmarks and consequent lack of milestones at which code is considered to be stable and released.

The challenge in developing and assessing benchmarks is to strike a correct balance between an application which adequately stresses the methods and tools, and one which does not exceed schedule and cost constraints. In

some respects, the necessity to complete a system on time and on budget tempered the effort of "Reinventing Electronic Design" in Benchmark-1.

8: Conclusions and Recommendations

The first Benchmark application thread has exercised the simulation methodologies of both developers and has been an impetus for development of improved methods. At Martin, it has also been used to demonstrate new software development methods. While the first RASSP benchmark emphasized architectural trade-offs and VHDL virtual prototyping, few tools are presently available to support VHDL virtual prototyping and co-simulation at the system level, and the methodology is still immature. Nevertheless, both Developers have now demonstrated VHDL virtual prototypes of different high-performance SAR processors. The Developers adopted significantly different approaches to modeling the processors, based in part on the COTS versus custom emphasis in the respective designs. Conclusions regarding the value of the modeling approaches await the development of the physical prototypes in Benchmark-2.

At the conclusion of Benchmark-2, it will be possible to compare the time and cost associated with the RASSP development process to that of a standard practice model based on parametric cost estimation. The expectation is that the effort invested in VHDL virtual prototyping will result in fewer defects in the physical prototype, and reduced effort during integration and test. An important by-product of the virtual prototype development is a comprehensive set of VHDL models which document the design at various levels of abstraction.

The first benchmark has also provided quantitative data on VHDL coding productivity and costs, and has highlighted areas for attention and improvement by the RASSP Developers, the technology-base contractors, and commercial EDA vendors. Several recommendations related to VHDL methodology are noted below:

- VHDL Modeling: As experience in board-level VHDL modeling is gained, methods for planning the effort should be developed and documented. Model-related issues include development of interface standards between models, selecting appropriate levels of abstraction, logical to physical mapping strategies, and interoperability of simulators.

- Estimating Effort: The immaturity of board-level VHDL modeling makes estimation of effort to model a particular system difficult. Chip designers tend to estimate in terms of gates and gates/day of productivity, but for board-level behavioral modeling, estimating the number of lines of VHDL and the lines-of-code/day is more appropriate. Calibration of VHDL productivity based on the benchmarks and demos is therefore important in developing a good estimation capability.

- Execution Time: There is a non-linear relationship between the fidelity of the VHDL models and the execution time. Careful selection of model abstractions and data sizes is essential in producing short simulation cycle times. However, this approach to controlling simulation time trades fidelity for speed and can result in certain types of design defects being missed in the virtual prototype. Methods of accelerating VHDL simulation are therefore important in realizing the full potential of VHDL in detecting and resolving defects prior to hardware fabrication and integration.

- Re-Use: The cost-effectiveness of VHDL virtual prototyping will improve significantly through the development of mechanisms for the re-use of VHDL models. Re-use encompasses not only library development and population, but also synthesis of VHDL code from system-level design tools.

Acknowledgments

We acknowledge the personnel on the Benchmark teams at Lockheed Martin - Sanders and Martin Marietta Laboratories, without whose cooperation this work would not have been possible.

References

1. G.A. Shaw, "RASSP Benchmark Program Overview," Proc. 1st Annual RASSP Conf., Arlington VA, Aug. 15-18, 1994, pp. 33-42.

2. B. Zuerndorfer and G.A. Shaw, "SAR Processing for RASSP Application," Proc. 1st Annual RASSP Conf., Arlington VA, Aug. 15-18, 1994, pp. 253-268.

3. B.W. Zuerndorfer, J.C. Anderson, R.A. Ford, A.H. Anderson, G.A. Rocco and G.A. Shaw, "RASSP Benchmark 1 Technical Description," Project Report RASSP-1, Rev. 1 (ESC-TR-95-001), MIT Lincoln Laboratory, Lexington MA, Jan. 25, 1995.

4. A.H. Anderson, G.A. Shaw and C.T.Sung, "VHDL Executable Requirement,'' Proc. 1st Annual RASSP Conf., Arlington VA, Aug. 15-18, 1994, pp. 87-90.

5. A.H. Anderson and G.A. Shaw, "RASSP Benchmark-1 Executable Requirements User's Manual," Project Report RASSP-2 (ESC-TR-94-114), MIT Lincoln Laboratory, Lexington MA, 20 December, 1995.

6. E. Rundquist (Sanders), ‘‘Virtual Prototyping of a Synthetic Aperture Radar Processor and RASSP Benchmark-1,” Proceedings of the Sixth IEEE Workshop on Rapid System Prototyping, Chapel Hill NC, 7-9 June 1995.

7. L. Lee (Sanders), Private Communication, June 1995

8. E. Rundquist (Sanders), Private Communication, April 1995.

9. N. E. Fenton, “Software Metrics,” Chapman & Hall, London, 1991.

10. R. Hillson, “Support Tools for the Processing Graph Method,” Proceedings of the Fifth International Conference on Signal Processing Applications and Technology (ICSPAT 94), Dallas TX, 18-21 Oct 1994, pp 756-761.

11. Lockheed Martin, “RASSP Benchmark 1 Final Review,” Martin Marietta Laboratories, 29 March 1995.

12. W.G. Kline (Martin Marietta Laboratories), Private Communication, May 1995.

13. J.C. Anderson, “Predicting the Future with RASSP Benchmarks,” Proc. 1st Annual RASSP Conf., Arlington VA, Aug. 15-18, 1994, pp. 278-282.

14. J.C. Anderson, “Projecting RASSP Benefits,” Proc. 2nd Annual RASSP Conf., Arlington VA, July 24-27, 1995.

15. J. Mastretti, M. Sturlesi and S. Tomasello, “Static Analysis of VHDL Code: Simulation Efficiency and Complexity,” VHDL International Users Forum, San Diego CA, 2-6 April, 1995