Attention: Please Read This Page

This paper was prepared in 1999. While we believe that the material is important and will be interesting to current users of Foresight Systems software, we have been through some changes since the paper was written.

Nu Thena Systems, Inc. was a predecessor company to Foresight Systems M&S. All references to Nu Thena are obsolete. Readers should assume that Foresight Systems M&S replaces all instances of Nu Thena Systems, Inc.

The current contact information for Foresight Systems M&S is:

Foresight Systems M&S
12550 N. 89th St.
Scottsdale, AZ 85260

http://www.foresightsystems-mands.com
fs_marketing@foresightsystems-mands.com

The current production versions of the software do not support the direct synthesis to conventional hardware description languages. **FS/CoderC++** does automatically translate graphical Foresight executable system models into C++ code, which can be compiled and linked with a run-time library to form stand alone executables.

The diagrams and illustrations included here were prepared with versions of the software current at the time that the paper was prepared. Some of them may not be identical to equivalent images from the current version of the software. We apologize for any potential confusion.

Please excuse the use of any out of date expressions and terminology. The authors believed that their application of technical terms was consistent with standard practice at the time that this document was written.

If you have questions, concerns, or issues, please contact:

Dean Stevens
President, Commercial Systems Group
Foresight Systems M&S
deans@foresightsystems-mands.com
(408) 278-3983

Designing Affordable Avionics Systems with Advanced Modeling and Simulation Tools

Michael D. Vertal, John E. Crowley
Nu Thena Systems, Inc.,
McLean, VA 22102

Nicholas J. Babiak
Cambridge Research Associates, Inc.,
McLean, VA 22102

Abstract

To meet the challenges of today's avionics designs and modernization programs, advanced systems modeling and simulation tools are available which allow designers to ensure the correctness of system functionality and to optimize avionics architectures. This paper describes a commercially available system design tool set, Foresight, which has been used for avionics systems development. Foresight distinguishes itself with a combination of three key advantages: 1) an intuitive, highly graphical systems modeling language for functional behavior modeling and analysis 2) system architectural modeling constructs that permit the design and optimization of avionics architectures, 3) automatic translators to support detailed component design, integration, and test of complex avionics hardware and software.

1.0 Introduction

The rapid advancement of implementation technologies such as VLSI circuits and broadband communications coupled with the evolution of avionics architectures continues to challenge today's avionics designers. New designs must incorporate the latest technologies to satisfy the increasingly demanding requirements placed upon avionics while meeting the increasingly restrictive size and power requirements. And existing systems must be retrofitted with the latest technologies to maintain the utility of the platform To maximize productivity and fully leverage the underlying implementation technologies, avionics systems designers are moving to higher levels of abstraction for their system design, capturing their designs with implementation-independent, executable models and then transforming the designs into detailed designs which incorporate the latest implementation technologies.

This concept of implementation-independent, executable specifications has successfully demonstrated benefits in a few specific domain areas [3]. For example, VHDL [8] is currently gaining wide acceptance for its ability to model and simulate digital electronic systems at multiple levels of detail, allowing designers to capture and analyze their designs at a behavioral level, and with the use of commercially available synthesis tools, automatically generate ASIC designs from high level descriptions. Another example includes feedback control system design. Commercially available tools allow control designers to model and analyze complex control systems, move freely between various domains (time, frequency) to design complex algorithms, and then automatically generate software code from the design. Extending these capabilities to the systems level, that is, above individual hardware components [9] and software components, would give today's avionics system designers a powerful weapon to address their complex design challenges.

In this paper, we describe an executable systems modeling language and its supporting toolset. Foresight [2], which allows avionics system designers to capture system requirements and design models, and dynamically analyze their system designs for functional correctness and to explore and optimize system architecture. Supporting tools then translate the system designs into domain-specific languages, such as VL for hardware and C for software, suitable for downstream component design. Foresight provides comprehensive and expressive modeling constructs to support a wide array of simulation models that are needed throughout systems development, including detailed functional modeling, dynamic behavioral modeling, architectural performance modeling, and information modeling; and, an integrated tool set for constructing and simulating system models, which also supports extensions of system models into operational prototypes. Section 2 provides an overview of Foresight's systems modeling language and supporting tool set. Avionics design examples with Foresight are then discussed in Section 3. Conclusions are drawn in Section 4.

2.0 Foresight Overview

The complexity of avionics systems drives their design towards a complex combination of distributed, concurrent, event driven and data driven processes implemented as an integrated collection of custom and of-the-shelf hardware

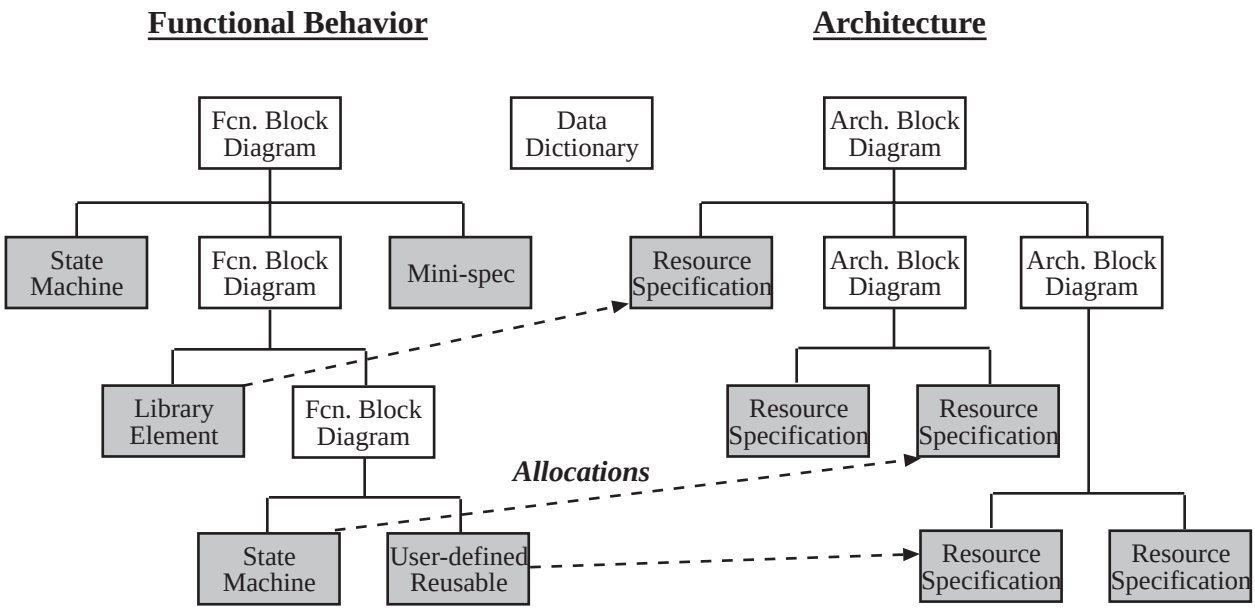

Figure 1. A Foresight Model Hierarchy

and software. To capture complex system requirements and designs into executable models, Foresight provides a rich, expressive modeling language that is highly graphical, precise, and machine executable. Foresight's graphical and textual modeling constructs allow system designers to capture a model of the most important characteristics of complex systems—their functionality, dynamic behavior, and architectural performance. Also, Foresight's data dictionary permits designers to capture information flow and interface requirements in the executable model.

Foresight's built-in support for functional, behavioral, performance, and information modeling in a graphical, intuitive language is unique and critical to support the important activities of systems development. Functional modeling constructs are necessary to ensure the correctness of function descriptions, not just the existence of the functions. Moreover, functional modeling facilitates important prototyping activities of systems development, which are moving further upstream in the systems development life cycle with distributed simulation technology [6]. Behavioral modeling constructs are required to capture the complex patterns of dynamic behavior of system functions, such as sequencing, concurrency, synchronization and fault detection/recovery. Performance modeling constructs are necessary to ensure the system architecture and components that perform system functions are sufficient to meet the non-functional requirements and design constraints such as queuing, throughput, timing, accuracy, and reliability. As multiple functions are often allocated to the same architectural components, contention for these resources must also be analyzed to eliminate bottlenecks and optimize system design. Finally, information-modeling constructs are required to easily and accurately capture system data requirements and interface requirements.

Foresight models are built hierarchically from a collection of modeling constructs. Foresight's primary modeling construct is a block diagram [1]. Block diagrams allow systems to be described as a network of communicating processes, which are well suited for modeling embedded system functionality and their inherent concurrency. Foresight's semantic extensions also permit modeling of control behavior, sequencing, and queuing. Furthermore, Foresight allows designers to capture designs in the most appropriate fashion for the system under design, since Foresight supports both an object-oriented approach [7] and a functional decomposition approach [4] [5]. As shown in Figure 1, the top level of any Foresight model is captured with a block diagram, which consists of an interconnected collection of process blocks. And the block diagrams can be used to capture both the functional behavior (functional block diagrams) and the architectural configuration (architectural block diagrams).

The functional block diagram process blocks may be decomposed with more functional block diagrams, or with Foresight primitive, executable modules (which are shown as shaded blocks in Figure 1): state machines, mini-specifications, library elements, or user-defined reusable elements. The primitive elements can capture functional descriptions, control logic, dynamic behavior, and performance parameters. For example, avionics designers may model function-

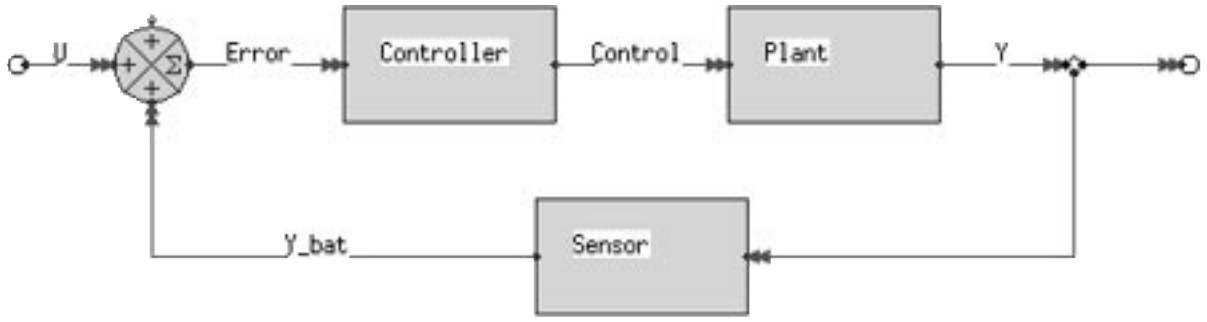

Figure 2. Foresight Functional Block Diagram Example - Feedback Control System

ality such as flight control laws, sensor management, and maintenance activities. Further, all process blocks are connected with information flows, whose information type specifications are stored and maintained in the data dictionary.

Similarly, architectural block diagrams may be decomposed to reveal additional architectural block diagrams, and at the primitive levels the performance characteristics of the components may be specified with resource elements. For example, avionics designers can specify buses with a certain bandwidth, and the CPUs with a certain MlPS rate.

3.0 Avionics Design Example Models

Cambridge Research has successfully applied Foresight to the development and verification of digital avionics systems. For example, an unmanned air vehicle verification facility was developed at Cambridge. Foresight was used to model the air vehicle's avionics, the payload and other subsystems, and the ground control station. Dynamic analysis with Foresight allowed verification of system designs and interface specifications.

3.1 Functional Verification

Foresight's functional modeling capabilities allow avionics designers to ensure the correctness of system functionality. As shown in Figure 2, Foresight modelers can model flight controllers and through time-domain simulation, ensure the stability and correctness of their designs. Often, Foresight is used in conjunction with control analysis packages such as MATLAB and MatrixX. After verifying the functional correctness of their designs, engineers can proceed to explore the architectural design with Foresight. That is, they may explore how the functional elements will be partitioned into hardware and software components, and how these components will be configured (software threads, boards communicating over high bandwidth. time-multiplexed buses, etc.). An example of functional behavior modeling of an avionics bus is shown in Figure 3, which illustrates a model of the HSDB (High-Speed Databus, also known as the Linear Token Passing Bus [10]). The 50 Mb/s HSDB is a Joint Integrated Avionics Working Group (JIAWG) standard designed for the next generation of military aircraft such as the F-22 Advanced Tactical Fighter and the RAH-66 Commence Helicopter, and can accommodate a range of applications ranging from large block data transfers down to low-frequency, periodic short message sequences.

The HSDB has the following characteristics:

- A token is used to determine which station has permission to talk on the bus.
- During start-up and reset conditions, a specific arbitration process is used to determine which station acquires the token.
- The logical ring is automatically constructed via sequential polling.
- The bus will automatically reconfigure if a new station comes on-line, or a station fails.
- The bus accommodates prioritized messages (priority 0 down to priority 3)

Foresight was used to model the functional behavior of the HSDB as defined by the JIAWG specification [11], in order to verify its correctness and to explore scenarios of its implementation within avionics architectures. For example, the model in Figure 3 illustrates a 10-station HSDB system.

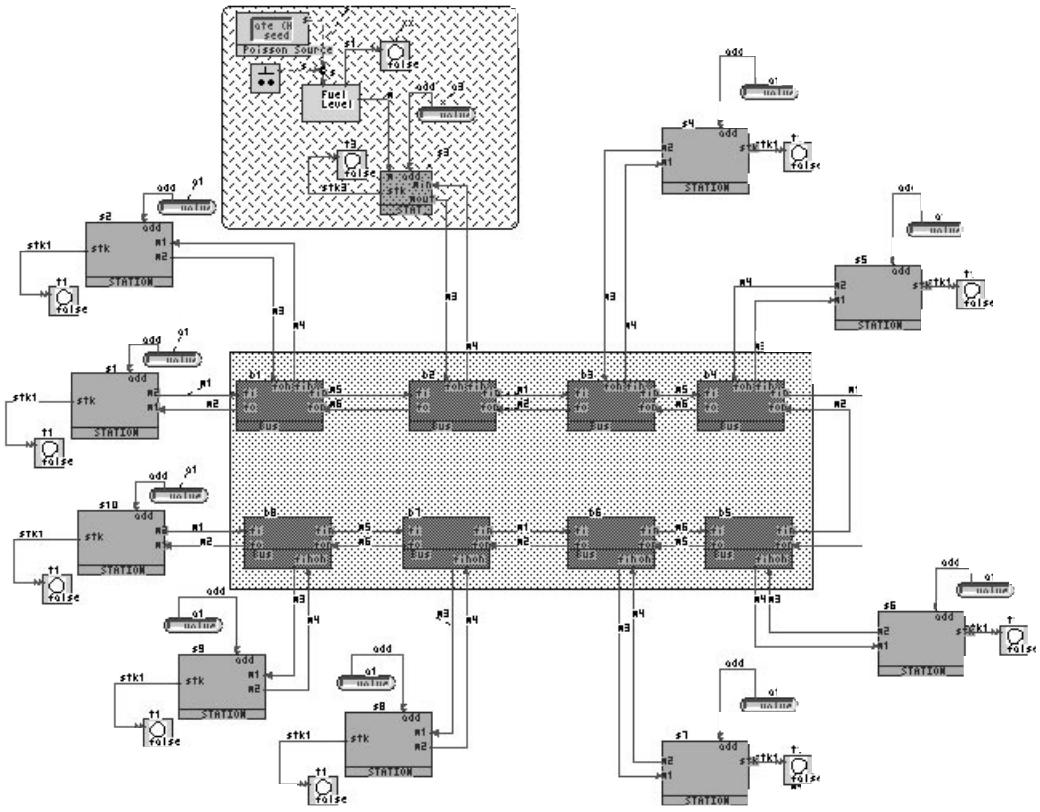

Figure 3. Foresight JIAWG High Speed Databus Example

This particular model was used to determine

1. message delays of low frequency (fuel level) messages on the bus, as a function of the message priority.
2. time required to construct the logical ring, and
3. start up behavior

Figure 5 illustrates the Foresight state machine that captures the top-level behavioral specification of the HSDB bus. A variety of bus data was rapidly derived from the Foresight model execution. The results from one particular scenario are shown in Figure 4; time delays for priority 0 messages ranged from 100 microseconds to 1 millisecond, whereas time delays for priority 1 messages ranged from 1 to 10 milliseconds. Another model execution demonstrated

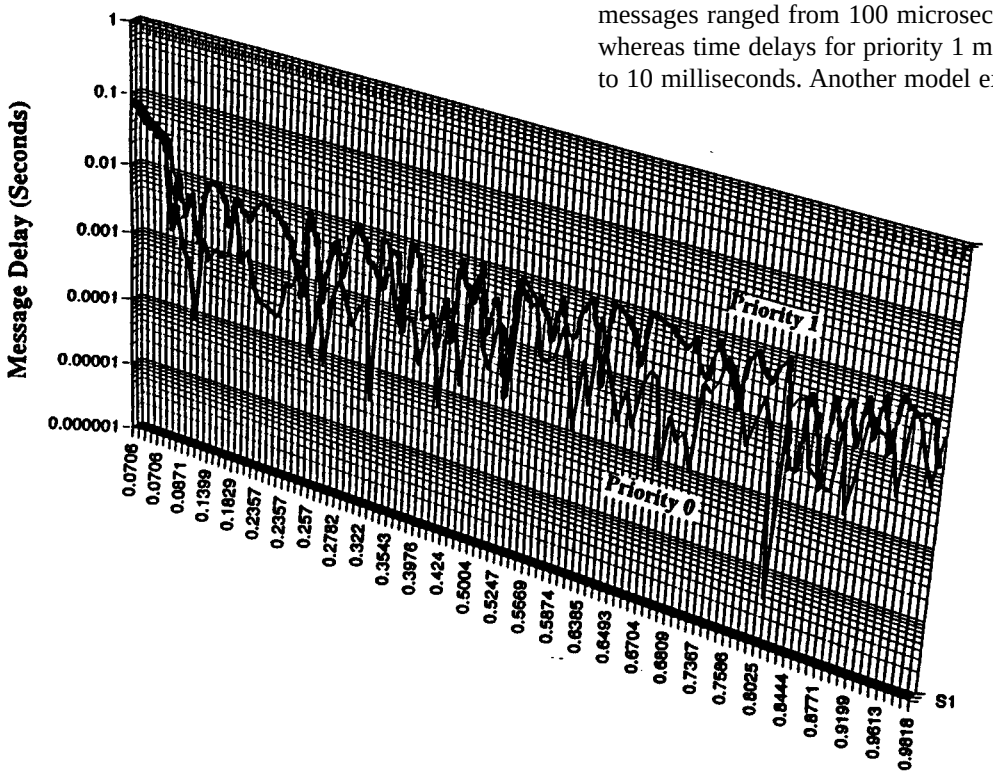

Figure 4. HSDB Model Execution Results - Message Delay

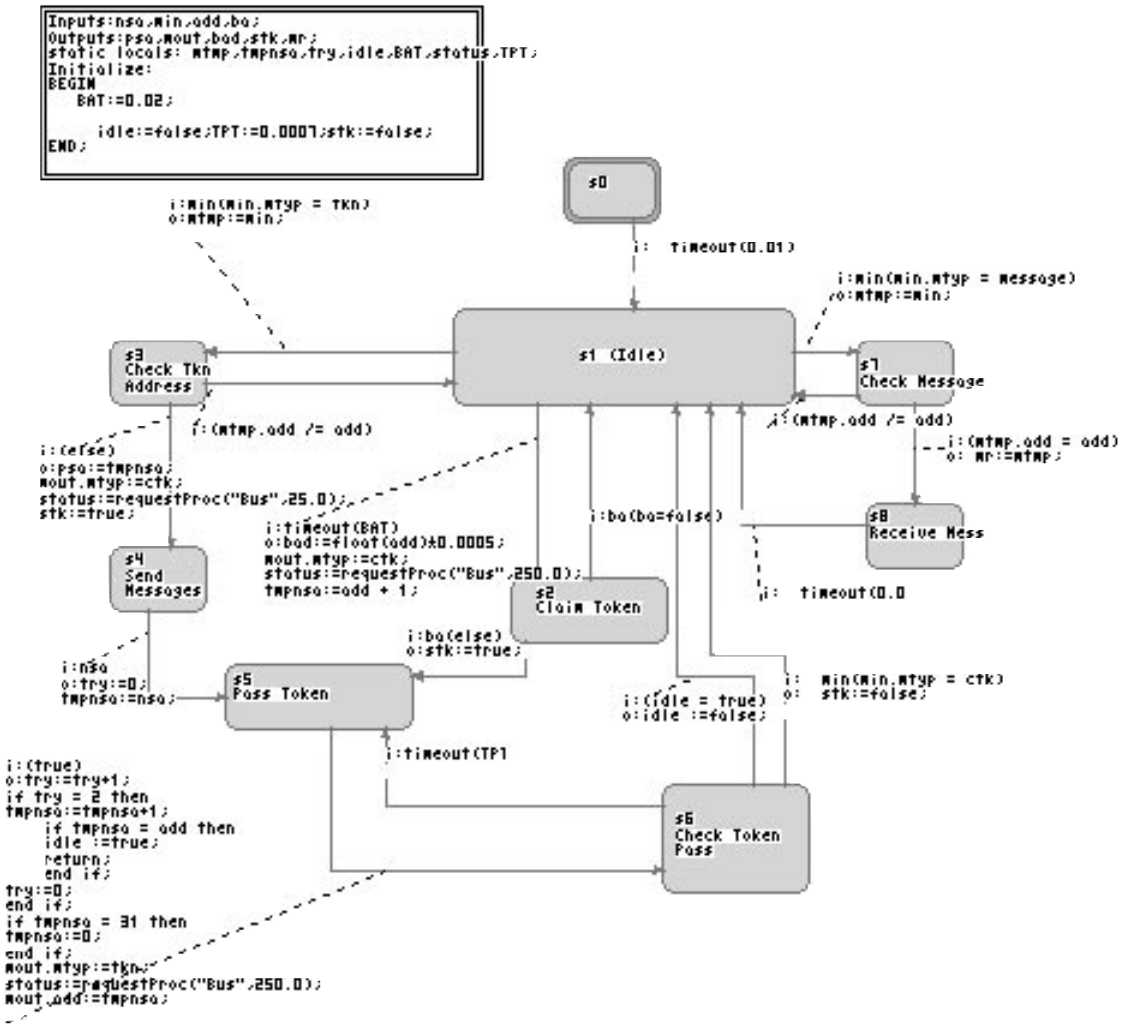

Figure 5. High-Speed Databus Behavioral Specification

that the time required to construct the logical ring during startup varied widely, and was proportional to the number of priority 0 messages on the bus during startup. The ability to perform this type of functional behavior analysis allows avionics developers to ensure that their designs are correct, before proceeding into detailed hardware and software design.

3.2 Architectural Design

Whereas the previous examples illustrate Foresight's functional behavior modeling capabilities that are used to verify functional designs, Foresight also supports the analysis, design, and optimization of system architectures. With these capabilities, avionics designers can allocate functional elements to architectural components such as CPU's, buses, memories, etc. These architectural components have performance constrains (e.g. MIPS for a CPU) which can be modeled and analyzed to optimize the avionics architectural design. Figure 6 illustrates an example of the avionics architecture of a military helicopter. The dynamic execution of the system model shows the utilization of the CPU and memory in this particular example.

3.3 Detailed Component Design

Following avionics system architectural design, detailed component design may proceed. To support the detailed design activities, Foresight incorporates import/export utilities and code generators to transition the system design to discipline-specific designs and tools. Foresight's export utilities permit system models to be represented in an ASCII format (based on the CASE Data Interchange Format) for import into software engineering (CASE) tools to support detailed software design. More significantly, a C code generator will generate C source code for all or a portion (e.g. the software partition) of the system model. Similarly for the digital electronics hardware design, a VHDL generator

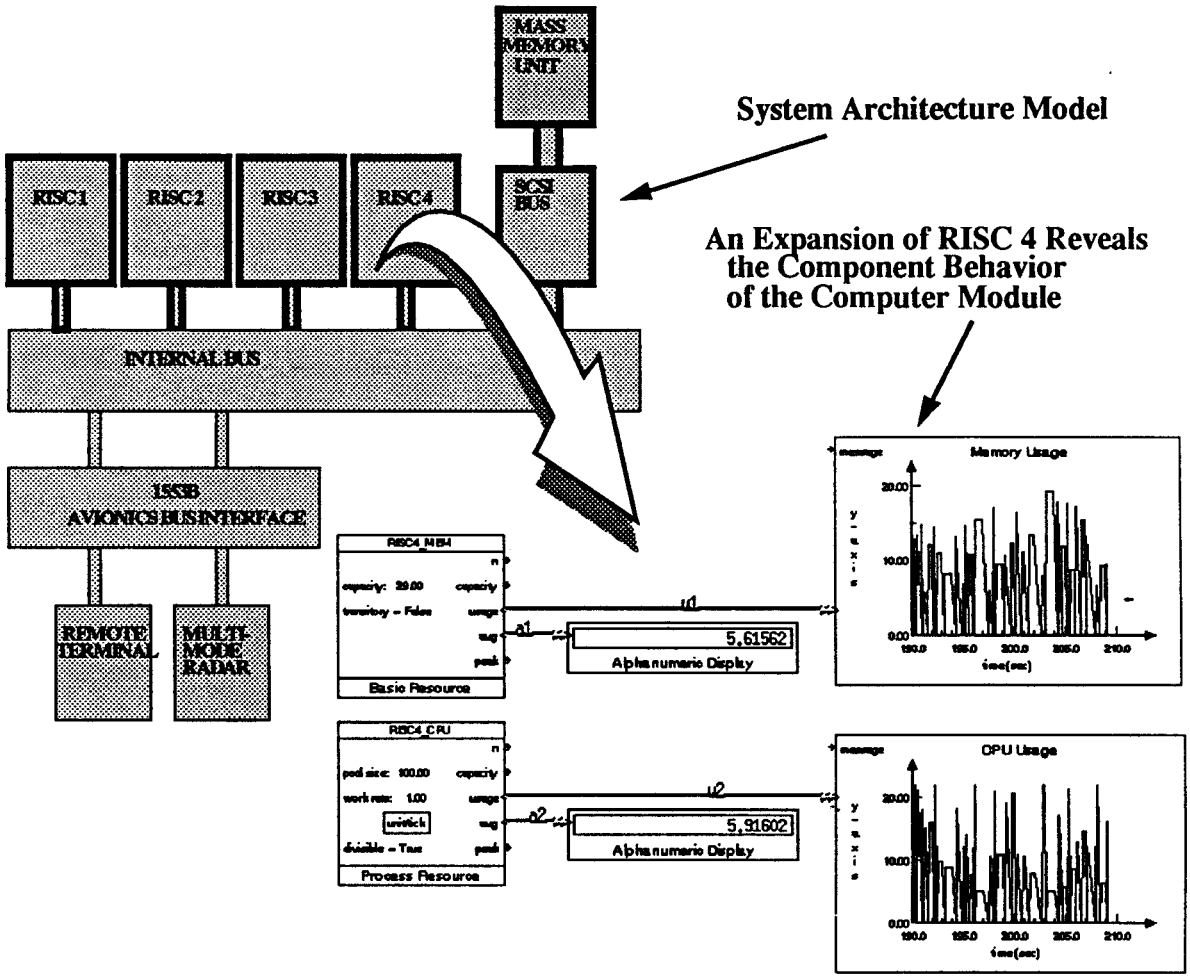

Figure 6. Avionics System Architecture Design

will translate the Foresight model directly into VHDL for use in both detailed board-level design as well as ASIC design Figure 7 demonstrates how Foresight models may be utilized to capture new designs as well as existing system designs (for modernization programs), and with automatic translators, provide a technology independent description of avionics systems to facilitate maintenance and upgrades throughout the entire life-cycle of a system.

4.0 Conclusion

Foresight provides avionics systems designers with a comprehensive system simulation tool set which can be used to significantly improve avionics system development.

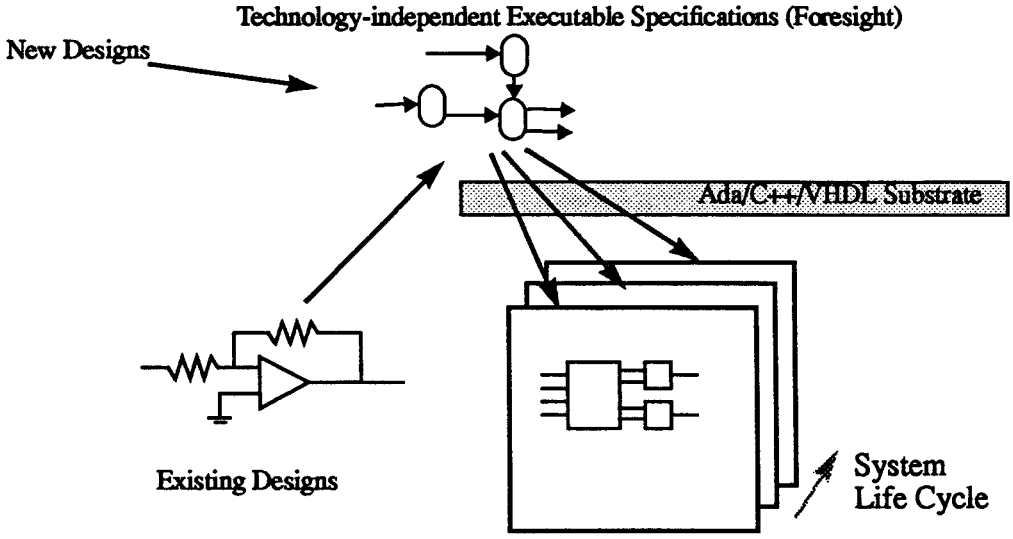

Figure 7. Automatic Translation for Component Design

Foresight executable system models are hierarchical, graphical models that fully capture the functional, behavioral, and performance characteristics of a system, along with its information flow and interface requirements. Foresight's modeling language is based on semantic-extensions to existing systems modeling notations, and it provides a tool set to easily build, analyze, and extend system models. Leveraging the full benefits of system simulation, avionics system developers can reduce requirements and design errors, and shorten development cycle times.

5.0 Acknowledgments

The authors are grateful to Mr. Jack Snow of Harris Corporation for providing insight into the high speed databus operation.

6.0 Trademarks

7.0 References

[1] B. Blanchard and W. Fabrycky, *Systems Engineering and Analysis*, Second Edition, Prentice Hall, 1990.

[2] M. Vertal, *Foresight: System Simulation for System Developers*, Proceedings of the 27th Annual Simulation Symposium, IEEE Computer Society Press, La Jolla, CA, April 11-14, 1994.

[3] P. Zave, "An Operational Approach to Requirements Specification for Embedded Systems," *IEEE Transactions on Software Engineering*, pp. 250-269, May 1982.

[4] D. Hatley and I. Pirbhai, *Strategies for Real-Time System Specification*, Dorset House, New York, 1987.

[5] P. Ward and S. Mellor, *Structured Development for Real-Time Systems*, Yourdon Press, Prentice Hall New York, 1985.

[6] M. Verbal, F. Adagio, and J. Garstka, "System Modeling for Distributed Interactive Simulation," Proceedings of the 1993 Southeastern Simulation Conference, Huntsville, AL, October 18-19, 1993.

[7] J. Rumbaugh, M. Blaha, W. Premerlani F. Eddy, W. Lorensen, *Object-Oriented Modeling and Design*, Englewood Cliffs, New Jersey Prentice Hall, 1991.

[8] *IEEE Standard VHDL Language Reference Manual*, IEEE Std 1076-1987, IEEE, New York, NY, March. 1988.

[9] M B. Srivastava and R. W. Brodersen, "Using VHDL for High-Level, Mixed-Mode System Simulation," *IEEE Design & Test of Computers*, pp. 31-40, Sept. 1992.

[10] R. Uhlhorn and J. Snow, "Civil and Military Applications of the High-Speed Databus", *Avionics*, March 1994.

[11] *Standard Joint Integrated Avionics Working Group Linear Token Passing Multiplex Data Bus Protocol*, JIAWG Document J88-N2, 22 Feb. 1992.