

System Design with Foresight Seminar Cellular Phone System Demo Model



Attention: Please Read This Page

This paper was originally prepared in August of 1997. 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Please excuse the use of any out of date expressions and terminology.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

System Design with Foresight Seminar

Cellular Phone System Demo Model

Nu Thena Systems, Inc.

August 1997

1.0 Functional Modeling and Verification

Foresight supports the creation and verification of functional requirements models. As shown in Figure 1. Foresight functional models are built hierarchically from a collection of modeling constructs. Foresight's primary modeling construct is a hierarchical block diagram. These block diagrams allow systems to be described as a network of communicating processes, which are well suited for modeling computer-based system functionality and their inherent concurrency.

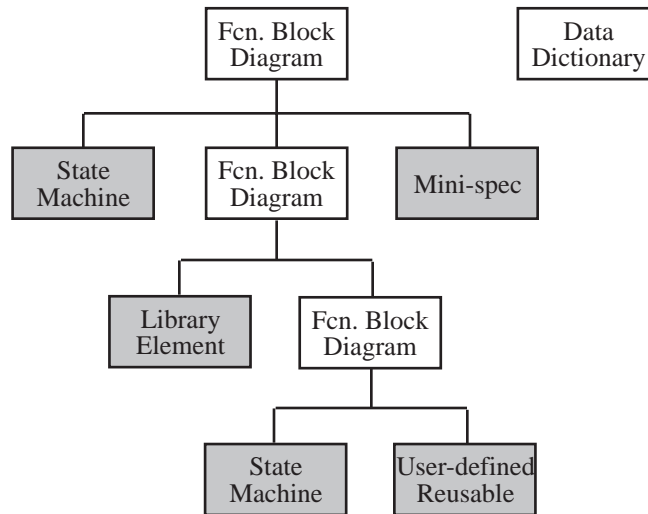


Figure 1. A Foresight Functional Model Hierarchy

Process blocks within a Foresight block diagram may be decomposed with more block diagrams, or with Foresight primitive, executable modules: state machines, mini-specs, pre-defined library elements, or user-defined reusable elements. These primitive elements have the expressive power to capture essentially any type of functional behavior, at any level of detail. All process blocks are connected with control and/or data flows, whose data types are stored and maintained in a data dictionary that supports a full range of types: bits, integers, reals, enumerated types, records, arrays, etc. Foresight's built-in mini-spec language, which is a procedural modeling language, has syntax similar to VHDL. This, combined with Foresight's familiar block diagrams and state diagrams enables engineers to quickly learn Foresight's comprehensive modeling language, and to rapidly exploit its expressive power. Additionally, Foresight's strong support for model reusability enables rapid model development and encourages design reuse.

An example block diagram consisting of two cellular phone interface models is shown in Figure 2.

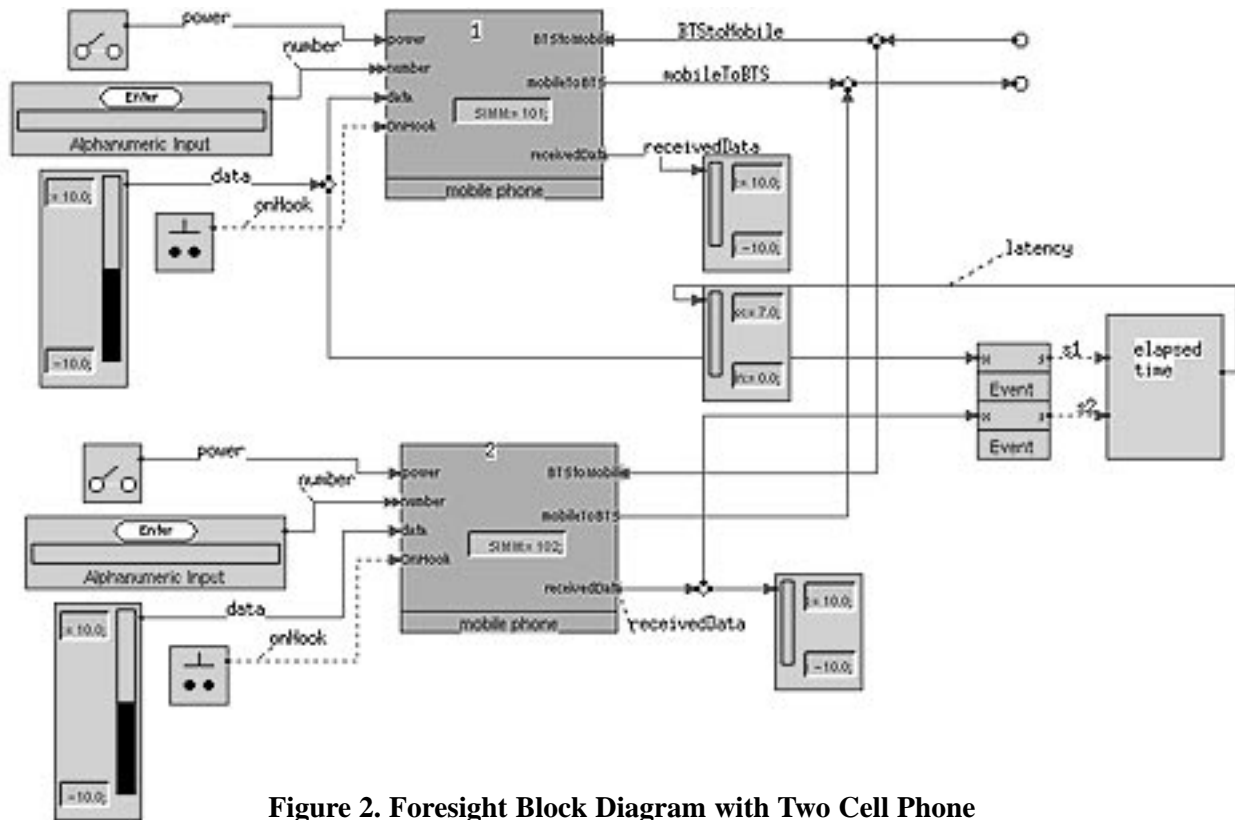


Figure 2. Foresight Block Diagram with Two Cell Phone

A state diagram of the cell phone's behavior is illustrated in Figure 3.

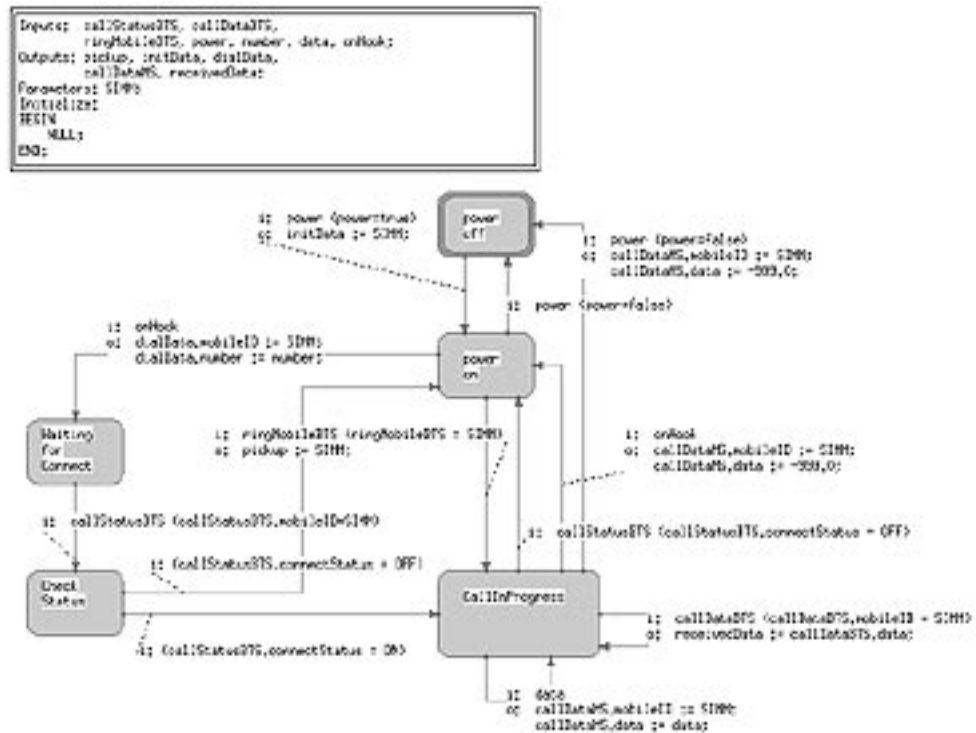


Figure 3. A Foresight State Diagram - Cell Phone Behavior

2.0 Architectural Modeling and Performance Analysis

Foresight supports high-level performance modeling with numerous built-in modeling constructs and libraries tailored for architectural performance analysis. Similar to other dedicated performance modeling tools and simulation languages, Foresight incorporates constructs such as queues with various scheduling mechanisms, resources, random number generators, etc. to easily capture high level architectural behavior. As illustrated in Figure 4, Foresight architecture models are similar to functional models in that they are built hierarchically, but at the lowest levels reside resource specifications for the architectural components.

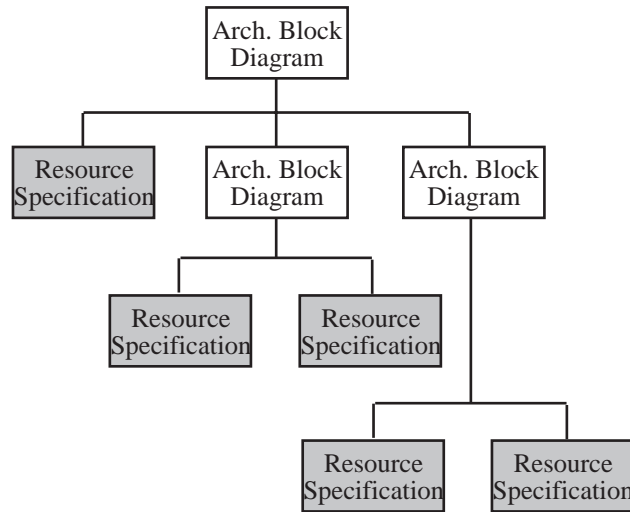


Figure 4. A Foresight Architectural Model Hierarchy

Figure 5 illustrates a Foresight architectural model of a cellular basestation's transmit/receive unit at the board-level.

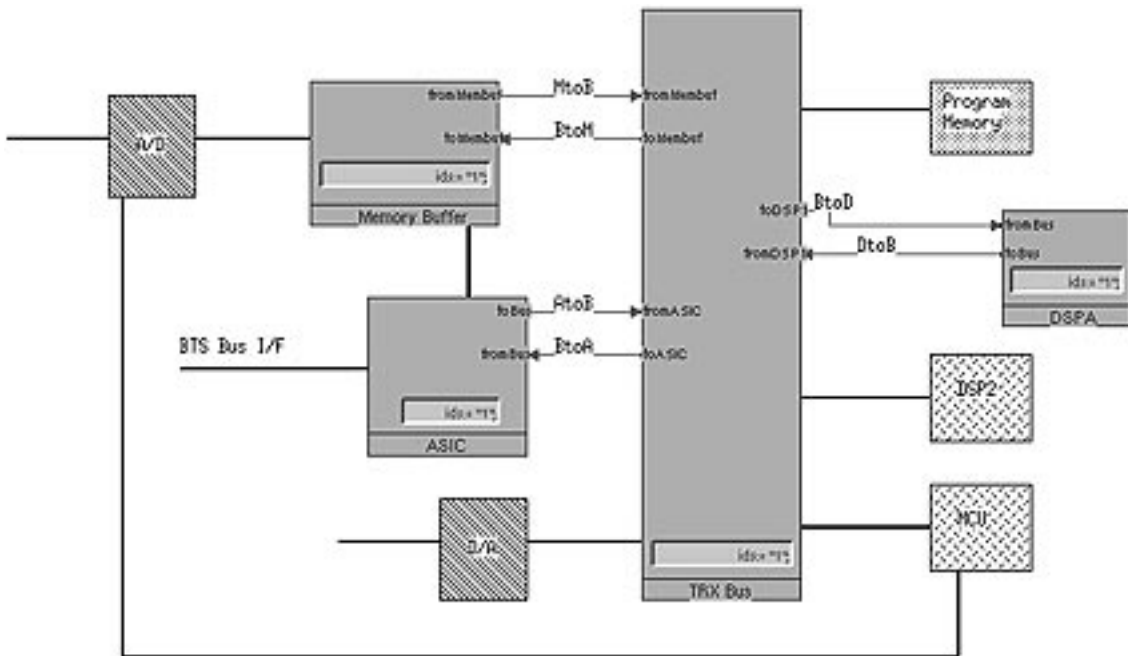


Figure 5. A Foresight Architectural Model - Cell Basestation Transceiver Unit

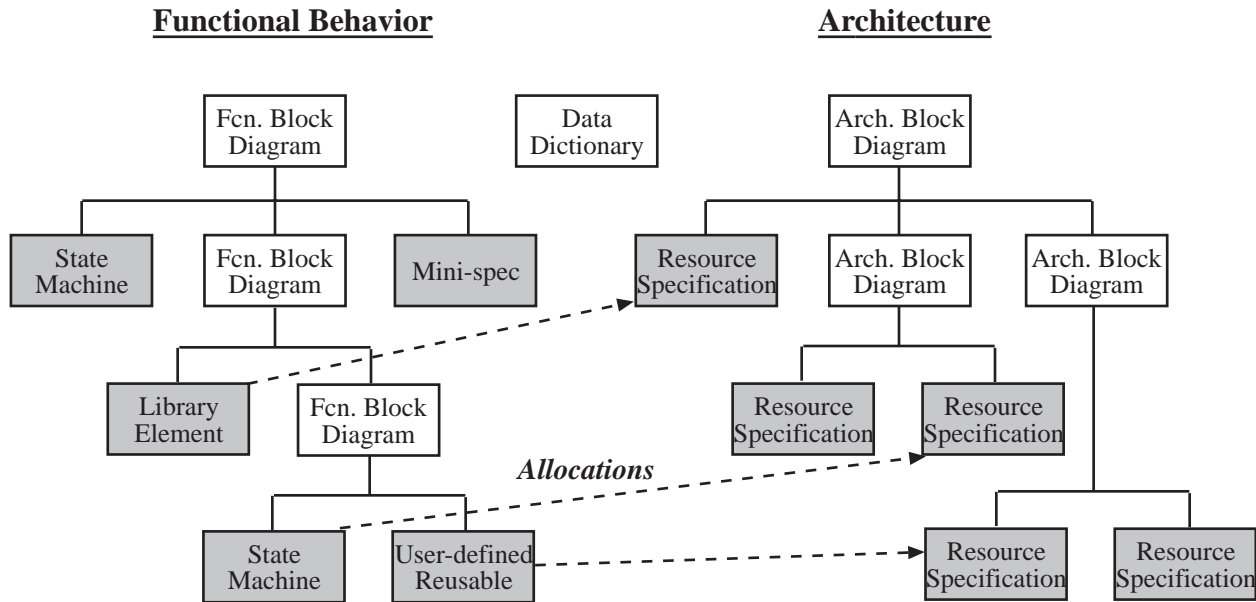


Figure 6. Foresight Integrates Functional Verification with Architectural Design

As shown in Figure 6, the two orthogonal views of the system — function and architecture — are linked by a partitioning specification that describes the mapping of functional objects onto architectural components. The combination of a functional model, an architectural model, and a mapping between them comprises a single system design alternative. Figure 7 illustrates a functional partition of a cell basestation’s signal processing functions onto the transceiver unit. The system simulation suggested an ASIC implementation for an *error correction*.

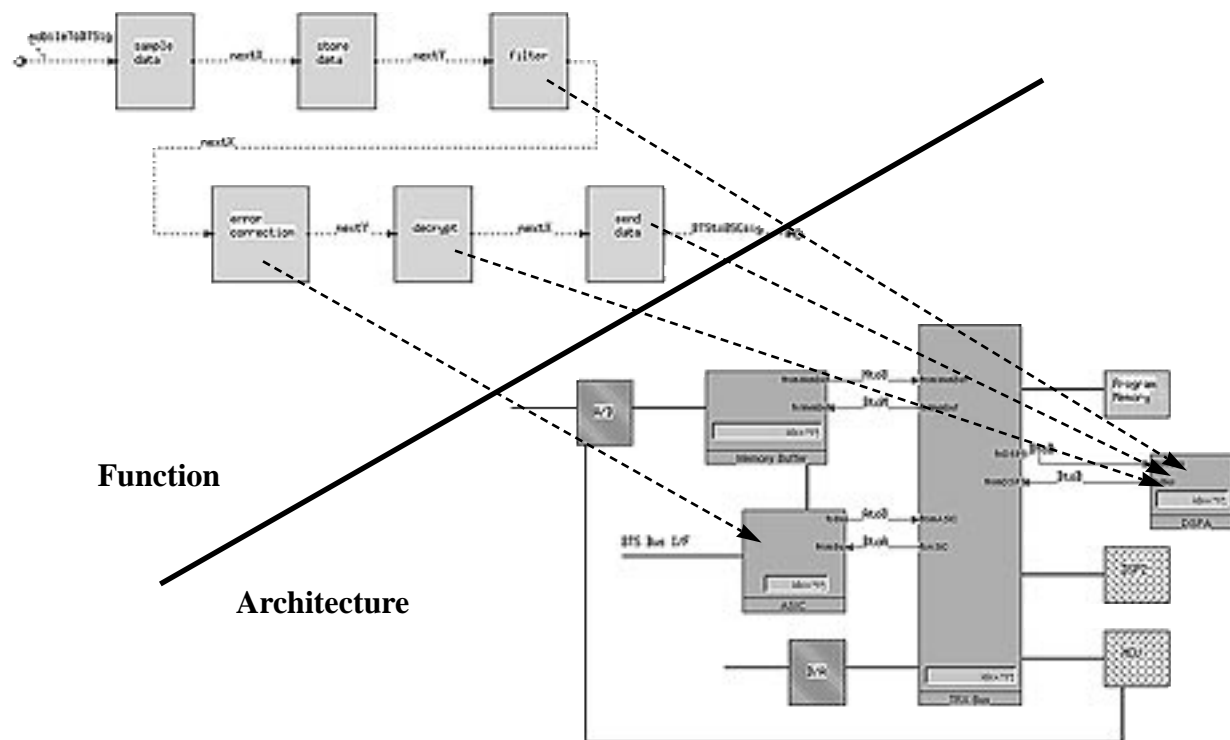


Figure 7. Foresight System Design - Hardware / Software Tradeoffs

3.0 Design Refinement and ASIC Testbench Generation

Foresight's functional modeling capabilities may be used to refine system design models to low levels of detail which when converted automatically to VHDL using FS/CoderVHDL may be used for ASIC design specification and testbench generation for downstream ASIC validation.

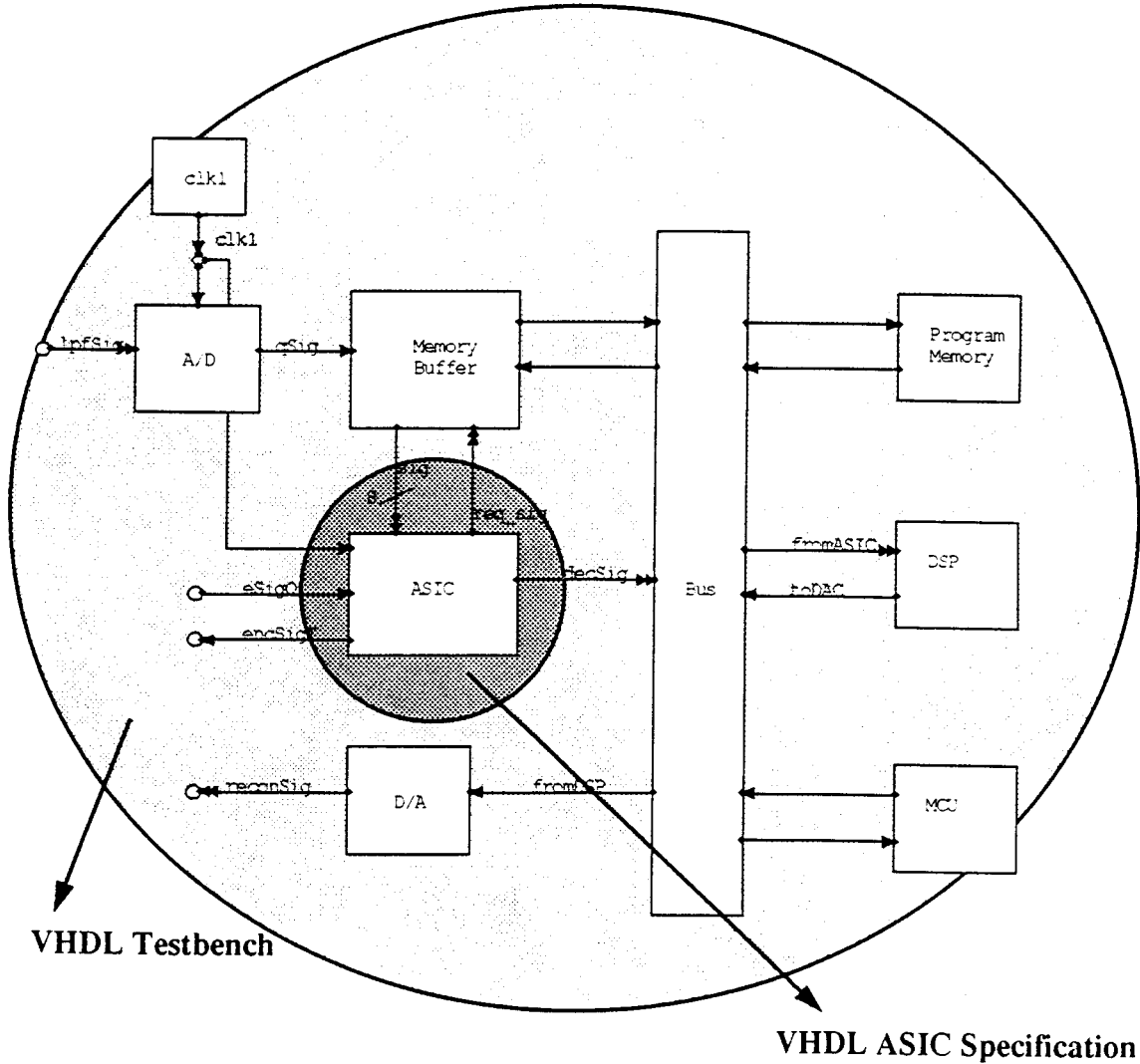


Figure 8. Foresight Design Refinement and VHDL Code Generation