System Design Using an Integrated Specification and Performance Modeling Methodology

Ambar Sarkar
Viewlogic Systems Inc.
293 Boston Post Road, Marlboro
MA 01752, U.S.A.

Abstract

A system-design methodology based on the synergistic integration of the specification modeling and the performance modeling design stages is presented. This synergy is supported by: 1) a novel technique that dynamically and automatically incorporates delay information into an executable specification from the corresponding performance model, and 2) a novel simulation-based algorithm that automatically checks conformance between the two models. VHDL is the primary means for integrating these two disparate design stages.

1. Introduction

As the design of a complex system proceeds from an abstract concept into a detailed design, significant time and effort is spent in analyzing the evolving stages. Communication between these stages is often awkward. Many design errors are introduced or left undetected in the final product due to miscommunication of design intent among these stages.

If not eliminated early, design errors typically manifest themselves during later stages when it is costlier to fix them. To support the early detection and elimination of these errors, a design methodology must provide effective communication among the various design stages of the product.

A review [2, 3, 17, 18, 19, 3, 1]of the state-of-the-art research in the field of digital-system design, however, reveals that support for interaction between the different design stages have not been extended to early design stages. Support for increased communication between levels of abstraction higher than algorithmic level is rare. For a detailed review of existing methodologies, see [10, 15].

The methodology presented here offers a strong interaction between models developed during the two early design stages: *operational-specification modeling* and *performance modeling*. Operational-specification modeling documents the external behavior of the system under design. The operational-specification model is an implementation-independent specification of the behavior, and is executable. A performance model abstracts the performance related features of an implementation. Qualitative and quantitative estimates of performance related characteristics are then obtained by a combination of analysis and simulation-based techniques.

There are three advantages of this novel methodology [12, 13, 15, 16]. First, a simulation-based technique, called *conformance checking*, detects conformance between a system specification and the performance model of its proposed implementation. The checking is automatic and requires minimal designer effort. Second, a novel technique, called *performance annotation*, incorporates implementation-dependent delays obtained from performance model into the specification model. The incorporation occurs automatically during the combined execution of both models. Since a specification typically lacks implementation-dependent information, such incorporation leads to more realistic simulation scenarios for debugging purposes. Third, a partial implementation can now be analyzed in the context of the specification of the rest of the system under design. The designer is therefore able to better study the suitability of the implementation compared to studying it only in isolation. Overall performance characteristics of the system can be predicted even if its perfor-

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00 ©1996 IEEE

```
Create test-bench in ADEPT
Create Integrated Model by linking:
    ADEPT test-bench
    Statecharts model
Simulate Integrated Model and obtain performance data
Repeat
    Select a component of the Statecharts model
    Create an ADEPT model for this component
    Update Integrated Model:
        Performance annotate Statecharts model component
        Link ADEPT model with Integrated Model
    Repeat
        Simulate Integrated Model.
        Check for conformance violations
        Correct Statecharts or ADEPT models if needed
    Until no more conformance violations occur
    Simulate Integrated Model and obtain performance data
Until the entire Statecharts model has been accounted for.
Connect all the ADEPT models that were developed.
```

Figure 1. Integrated design methodology.

mance model has been only partially developed.

The specific modeling environments chosen are Statecharts[4] for operational-specification modeling and ADEPT [1] for performance modeling. Statecharts is a graphical language for describing system behavior and is an extension of the finite-state machine formalism. ADEPT is a simulation-based environment, utilizing the VHDL [5] language and has an underlying formalism based on Colored Petri Net[8]. It offers designers the ability to model the information flow of a proposed implementation, at a high level of abstraction.

The integration between operational specification and performance modeling is implemented using VHDL. The primary reason for choosing VHDL is its suitability for representing and analyzing different modeling domains and abstractions of a digital system. Tools are readily available for translating Statecharts and ADEPT descriptions into equivalent VHDL descriptions [7, 11]. Once the models are translated, their interactions with the environment can also be described using VHDL, allowing the execution of these very disparate models in a common simulation environment. Further, the designers can operate at the modeling language of their expertise, without requiring any extensive knowledge of VHDL.

The remainder of the paper is organized as follows. In Section 2, the design methodology is presented, followed by brief descriptions of the techniques of performance annotation and conformance checking. In Section 3, we briefly describe an application of the methodology for the design of a realistic and complex system. In Section 4, we report the results that were obtained from this application. In Section 5 we conclude with lessons learned and future work.

2. Integrated methodology

The integrated methodology is shown in Figure 1. First, a Statecharts specification of the entire system under design is developed. Next, a test-bench is developed using ADEPT. This test-bench emulates the environment in which the system under design is expected to operate.

The next step is to link the ADEPT test-bench with the Statecharts specification so that both models can be simulated concurrently, possibly exchanging simulation stimuli with each other. Linking is further described in Section 2.1. Collectively, this linked model is called the *Integrated Model*. Simulation of the *Integrated Model* at this stage allows the designer to obtain preliminary performance estimates without having developed any performance models. Note that the simulation occurs in VHDL environment, as explained further in Section 2.1.

It is desirable to develop complex designs in a stepwise-refined, incremental manner. Thus, the Statecharts description of the entire system is partitioned into several components. The designer's goal is now to develop implementations for each Statecharts component one at a time.

For each Statecharts component, the designer first proposes an implementation and develops a corresponding ADEPT model. To incorporate implementation-dependent delays specified in the ADEPT model into the Statecharts model, the latter is instrumented using performance annotation (See Section 2.2). As a result, the Statecharts model will execute with delays as predicted by the ADEPT model. The ADEPT model is then linked to the *Integrated Model*.

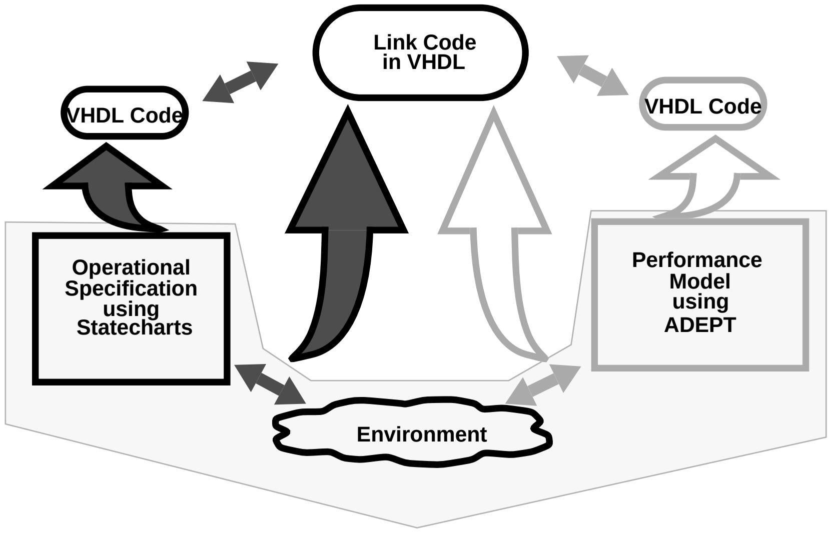

Link Code in VHDL
VHDL Code
VHDL Code
Operational Specification using Statecharts
Performance Model using ADEPT
Environment

Figure 2. Integrated Simulation of Statecharts and ADEPT model.

This newly linked *Integrated Model* is simulated next. Conformance errors are detected at this stage. These errors may be due to a combination of erroneous implementations or ambiguous specifications. The Statecharts and the ADEPT models are modified to eliminate these errors. The linking and simulation steps are repeated until the outputs of the two models conform. Once all conformance violations are eliminated, the designer may further refine previously obtained performance estimates to reflect the inclusion of the new ADEPT model.

Once all the components of the Statecharts model of the entire system have a corresponding conformed ADEPT model, all the ADEPT models are connected together. This ADEPT model now represents a conformed model of the entire system under design.

2.1 Linking for integrated simulation

Figure 2 shows how VHDL is used to simulate Statecharts and ADEPT models in the common environment. Linking is the process by which the designer specifies the interactions of a Statecharts or an ADEPT model with its environment. We call the act of specifying these interactions *linking* and the code emulating these interactions the *link code.*

Linking is specified by the designer in the model's language. The linking information can then be automatically translated into VHDL. A Statecharts or an ADEPT model can interact with its environment in a fixed number of ways. For example, a Statecharts communicates with its environment using events and conditions, whereas the ADEPT model communicates by activating or deactivating its ports. For each possible type of interaction of these models with their environments, we have defined rules to generate the corresponding link code in VHDL. Once all the interactions are specified, the link code can be automatically generated using these rules.

Both the Statecharts and the ADEPT model are translated into their equivalent VHDL code. The models are represented as separate VHDL components. The corresponding entity declaration of these VHDL components represent the interface through which these models exchange information with the outside world. This interface is a collection of VHDL ports. A stimulus generated by the model for its environment will appear as a event on the signal associated with the corresponding port. Similarly, any stimulus can be passed into the model by generating a VHDL signal on the corresponding port.

The actual exchange of information is managed by a separate VHDL component. This VHDL component contains the link code described earlier. The link code monitors VHDL events occurring in ports of one model and performs the task of maintaining interaction of the model with its environment.

2.2 Performance annotation

A Statecharts model typically lacks implementation-dependent delay information. Performance annotation instruments a Statecharts model so that delay information from the performance model is automatically incorporated into the execution of a Statecharts model. The execution of the Statecharts model and the performance model is synchronized. The modifications cause a Statecharts event to be delayed by the same amount of simulation-time as is encountered by the performance model for the analogous event.

The performance-annotation technique does not introduce any implementation dependent changes into Statecharts and preserves the design intent of the specifier. A complete description of this technique can be found in [10].

2.3 Conformance checking

A conformance error is detected when the ADEPT model produces an output that was either not predicted by the Statecharts model or had occurred in an order that violates the dependencies in the Statecharts. Two outputs have an output dependency if the occurrence of one output depends on the occurrence of the other output. The outputs generated by the two models are monitored and compared during simulation to see if they agree with each other. The conformance checking algorithm ensures that the dependencies between the Statecharts outputs are preserved in the set of analogous outputs produced by the ADEPT model and reports when these output dependencies are violated.

However, detecting violations of output dependencies is difficult. Corresponding outputs of the two models may be generated at different times, and possibly in different order. Note that two outputs in the Statecharts specification may have no dependencies on each other. This implies a non-determinism in the order of the occurrence of these outputs. If the corresponding ADEPT model produces outputs in a different order, that should not be flagged as a violation of conformance. However, if the outputs had dependencies, the ADEPT model should produce the outputs in the same order as produced by the Statecharts model. This problem is similar to that of validating circuits synthesized in the *free-floating IO scheduling mode* [20].

Conformance checking is similar to *comparison checking*[21] used in the context of *software-design diversity* [22], *back-to-back testing* [23]. However, none of these approaches allow non-determinism in their output sequence, and thus can severely restrict the design space by eliminating many designs even if they conform by our definition. A complete description and proof of this technique can be found in [10].

3. Token ring: an application

The design of a token-ring[6] was chosen as an example. The token-ring is a complex system that would be able to demonstrate the viability of the proposed methodology for realistic design problems.

The methodology was applied as follows. First a complete Statecharts description of the token ring was developed. A test-bench was developed in ADEPT to drive the simulation of the entire Statecharts model. These two models were linked together and executed to generate preliminary performance estimates.

A component of the token ring was identified next and an ADEPT model of that component was developed. This ADEPT model was then linked with the existing Statecharts model. Linking the ADEPT model effectively replaced the corresponding component in the Statecharts model of the entire token ring. This linked model was then simulated to collect updated timing information and to check for design or specification errors. If the timing results were not satisfactory or if there were remaining conformance errors, either the Statecharts or the ADEPT or both models were modified. This process was repeated until no more conformance errors were detected and the timings obtained were satisfactory. These steps were repeated for each component of the token ring.

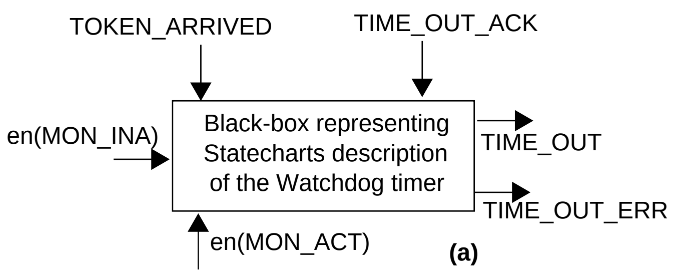
For a complete description of the application of this methodology to the design of token-ring see [9].

3.1 Designing the watchdog timer

One component of the token ring is the watchdog timer, which generates a timeout when it fails to detect any activity in the ring for a specified length of time. A black-box representation of the Statecharts and the corresponding ADEPT model for the watchdog timer is shown in Figure 3.

3.2 Simulation and conformance checking

Both the Statecharts and the ADEPT models were



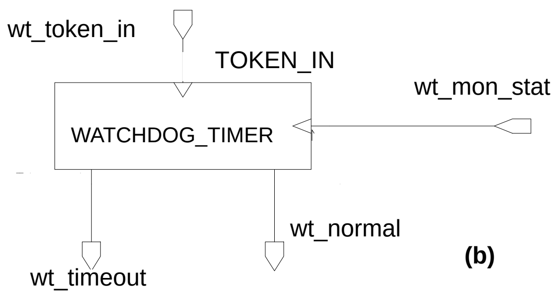


Figure 3. Identifying the interfaces (a) Blackbox representation of Statecharts model (b) ADEPT component

```
entity WATCHDOG_TIMER_S is
port (
      EN_MON_ACT:        in      trigger;
      EN_MON_INA:        in      trigger;
      TOKEN_ARRIVED:     in      trigger;
      TIME_OUT_ACK:      in      trigger;
      TIME_OUT:          buffer trigger;
     TIME_OUT_ERR: buffer trigger
);
end WATCHDOG_TIMER_S;                    (a)
```

```
entity WATCHDOG_TIMER_A is
port(
    WT_TOKEN_IN:          inout Token_res ;
    WT_TIMEOUT:           inout Token_res ;
    WT_NORMAL:            inout Token_res ;
    WT_MON_STAT:          in Token
 );
end WATCHDOG_TIMER_A;                   (b)
```

Figure 4. VHDL interfaces of a) Statecharts and b) ADEPT models of the Watchdog timer.

translated into VHDL components. The VHDL code corresponding to the Statecharts model of the watchdog timer is automatically generated by iLogix expressVHDL toolset[7]. Similarly, the ADEPT model was automatically converted to VHDL by the ADEPT tool. The entity descriptions of the two models are shown in Figure 4.

Figure 5 tabulates how events occurring on the interface of the watchdog timer relates to events for the corresponding Statecharts and ADEPT models. This information is provided by the designer. Generation of the link code can be automated, given information in this table.

The link code is created as another VHDL component that manages the interaction of the testbench with these models. Finally, the integrated model, including VHDL components representing the ADEPT and Statecharts models and the link code were simulated.

A commercial simulator was used to execute the VHDL code representing the integrated model. In this case, we especially monitored the TIME_OUT_ERR event. Generation of the TIME_OUT_ERR event indicated a conformance error between the Statecharts and ADEPT model of the watchdog timer.

4. Results

We applied our integrated-simulation based methodology to the design of a network based on IEEE 802.5 token-ring specification. The token-ring provided a nontrivial test-bed to demonstrate the effectiveness our integrated-simulation based approach. We were able to obtain several interesting results, which can be grouped into two major categories.

In the first category, a number of design errors were detected, which covered both specification and implementation errors. The sources of these errors were counterintuitive specification semantics, designer oversight, specification ambiguity, and misinterpreted design intent. The diversity in the range of sources of errors detected demonstrates how our methodology effectively uncovers a wide class of errors. In the second category, preliminary performance estimates were obtained using partially developed performance models, which may not have been feasibly obtained without integrated simulation.

The errors detected and performance estimated above might have been obtained independently without applying the integrated simulation approach. However, we are not aware of any other approaches that obtain such results in a reasonably efficient and practical manner. Using integrated simulation, we were able to detect errors that we believe would have otherwise propagated to lower-level design stages, or would have been left undetected.

Activity in Environment	Corresponding Statecharts Activity	Corresponding ADEPT activity
The token arrives at the station	generate event TOKEN_ARRIVED as input to Statecharts model	activate wt_token_in port
Monitor becomes inactive	EN (MON_INA) occurs as input to Statecharts model	deactivate wt_mon_stat port
Monitor becomes active	EN (MON_ACT) occurs as input to Statecharts model	activate wt_mon_stat port
Time-out occurs in ADEPT	TIME_OUT_ACK generated as input to Statecharts model	wt_timeout port activated

Figure 5. Identifying the interactions of between the models and their environments.

5. Conclusions and future work

Two important, early, but dissimilar design-stages are operational specification and performance modeling. We have shown how our integrated-simulation based methodology successfully reconciled the dissimilarities between these two design stages and produced a seamless environment for system design and verification.

The methodology was developed on the platform of Statecharts and ADEPT environments. The ideas developed here can be extended to other modeling languages and environments. In our approach, we perform the simulation of the integrated model in a VHDL environment. It would be interesting to observe the effect of the simulation on the Statecharts component of the integrated model in the Statecharts modeling environment. This way, a the designer will have a visual feedback of the effects of incorporating ADEPT related simulation information into the Statecharts model.

The ADEPT model can also be translated into an analogous Statecharts representation. This will allow one to apply analytical techniques from the Statecharts modeling domain to the ADEPT models. In fact, we have developed rules to translate an ADEPT model to Statecharts. These rules are presented in [10]. Further research is required to identify the advantages of such cross-domain analyses.

6. Acknowledgments

I would like to thank Ron Waxman and Jim Cohoon at the University of Virginia for their constant feedback and constructive criticisms. Thanks are also due to Samuel Sortais, Sylvan Revel, and Alain Gonier from IRESTE for their invaluable help in demonstrating various steps of the methodology.

7. References

[1] J.H. Aylor et al. The Integration of Performance and Functional Modeling in VHDL. In *Performance and Fault Modeling with VHDL*. Schoen, J. M., Prentice Hall, Englewood Cliffs, NJ 07632: 22-45, 1992.

[2] J. P. Calvez. Embedded Real-time Systems: A Specification and Design Methodology, *Wiley Series in Software-Engineering Practice,* 1993

[3] D. D. Gajski, N. Dutt, A. Wu. and S. Lin. *HIGH-LEVEL SYNTHESIS: Introduction to Chip and System Design*. Kluwer Academic Publishers, 1992.

[4] D. Harel. On Visual Formalisms. *CACM* 31:514-530 1988.

[5] IEEE. *IEEE Standard VHDL Language Reference Manual*. IEEE Inc., NY, 1988.

[6] IEEE. Token-Ring Local Area Network: Premier Issue, *IEEE Network*, Jan 1987, Vol. 1, No. 1.

[7] i-Logix Inc., *ExpressVHDL Documentation,* 1992

[8] J. Peterson. Petri-nets. *ACM Computing Surveys* 9(3):223-252 Sep 1977.

[9] S. Revel. *Integration of Specification and Performance Models.* IRESTE 3, July 1994.

[10] A.Sarkar, Integrating Operational Specification with Performance Modeling for Digital-System Design, *Ph.D. Thesis*, May 1995.

[11] S. Srinivasan. ADEPT: An Advanced Design Environment Prototype Tool. In *M.S. Thesis*., Department of Electrical Engineering, University of Virginia, 1990.

[12] A. Sarkar et al. Integrating Operational Specification and Performance Modeling. Fall'92 VHDL International Users' Conference, Washington DC, 10/18/92.

[13] A. Sarkar et al. System Design Utilizing Integrated Specification and Performance Models. *Proceedings, VHDL International Users Forum*, Oakland, California, May 1-4, 1994, pp 90-100.

[14] S. Revel. *Integration of Specification and Performance Models.* IRESTE 3, July 1994.

[15] A. Sarkar et al, A survey of specification methodologies for reactive systems, *Current Issues in Electronic Modeling, Issue 3,* Kluwer Academic Publishers, 1995.

[16] A. Sarkar et al. Integrating Operational Specification with Performance Modeling for Digital-System Design. *Current Issues in Electronic Modeling, Issue 6,* Kluwer Academic Publishers, 1996.

[17] D. Schefström and D. van den Broek, *Tool Integration: Environments and Frameworks*, Wiley Series in Software Based Systems, 1993.

[18] A. L. Opdahl and A. Solvberg. A Framework for Performance Engineering during Information System Development. *Advanced Information Systems Engineering (Advanced Information Systems Engineering CAiSE `92, Manchester, UK)*:65-87 May 12-15, 1992 proceedings.

[19] A. M. Davis. A Comparison of Techniques for the Specification of External System Behavior. *Communication of the ACM* 31(9):1098-1115 September 1988

[20] D. Knapp, T. Ly, D. MacMillen, and R. Miller. “Behavioral Synthesis Methodology for HDL-Based Specification and Validation.” *DAC-95*:286-291, Jun 1995.

[21] S. S. Brilliant, J.C. Knight and P.E. Ammann. On the Performance of Software Testing using Multiple Versions. *FTCS 20* pp. 408-415. 1990.

[22] C.C. Ramamoorthy, Y.R. Mok, F.B. Bastani, G.H. Chin, and K. Suzuki. “Application of a Methodology for the Development and Validation of Reliable Process Control Software”, *IEEE Transactions on Software Engineering,* Vol SE-7, No. 6, November 1981.

[23] M.A. Vouk, M.L. Helsabeck, K.C. Tai and D.F. McAllister. “On Testing of Functionally Equivalent Components of Fault-Tolerant Software”. *Proc. COMPSAC 86,* 1986, pp 414-419.