Proceedings of the 1986 Winter Simulation Conference
J. Wilson, J. Henriksen, S. Roberts (eds.)

DIGITAL SYSTEMS DESIGN AUTOMATION: A SIMULATION-BASED FRAMEWORK

Stephan Zorn
Siemens AG, Corporate Laboratories
For Information Technology, ZT ZTI
Otto-Hahn-Ring 6,D-8000 Munich,FRG

ABSTRACT

In this paper a system design environment is presented for the design of digital systems on higher abstraction levels. The higher levels begin with the requirements and go down to the register transfer level, where the use of existing CAD-systems can begin. In the first version a fixed method is offered that can be changed and adapted in later versions; in this ease the system design environment presents only a framework.

The most important step in our system design environment is the validation of a design. For this step several methods are offered: Petrinets, analytical and numerical methods and the most significant simulation.

After presentation of the methodology the tool support for the method is discussed. Tools which already exist and in addition newly developed tools are presented as well as their connections and combinations. In addition to the tools, some general sections describing a user interface and the data base are presented.

1. INTRODUCTION

Motivated by the desire to develop a complete computer system we decided to build a system design environment that supports digital computer systems design, concentrating on the higher design levels. At these levels the complete computer becomes visible, but no computer design systems are available. Here the most fundamental decisions must be made about the system. To do this you need tools to evaluate the alternatives; this is mostly done by simulation. But due to the increasing complexity of the systems being designed, it has become a real information problem to manipulate all of these data.

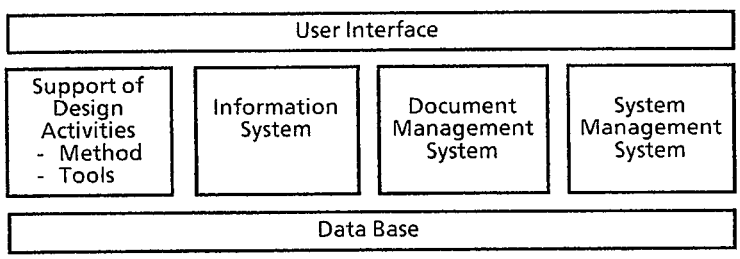
To handle this situation we have tried to offer more security in the design of computer systems. This will be achieved by offering a system design environment that contains a design method as well as tools to support it. The main parts are requirement analysis, design specification, design analysis, evaluation of the results, and the storage and retrieval of all the design data.

In the first version we present a fixed combination of tools. Following versions will present a flexible framework in which user specific tools may be integrated and the method can be adapted to given organisations. But the kernel for the design process remains always stays the design validation and this is done by simulation.

2. STRUCTURE OF THE SYSTEM DESIGN ENVIRONMENT

The system design environment consists of 6 functional subsystems (see figure 1).

- the user interface:
  This is the most general facet of the system. It exists for all subsystems and it provides a uniform interface to all the tools that are contained in the system. The total uniformity is not however within the first version. It also guides the user through the system. A special part of the user interface is reserved for a system manager. In this section all the commands for adapting the system design environment to special user requirements are presented.



System Design Environment: Functional Subsystems

figure 1

- the support of the design activities:
  This is the nucleus of the system. It contains the design method together with the tools supporting it. In later versions it will be possible to integrate special tools. Thus the system will be a design frame in which special tools may be integrated according to a user's special requirements.

- the information system:
  The information system presents information about the actual contents of the project and standard libraries. The user guidance utility of the user interface procures information that is needed to help a user: information about the tools, their use and the next steps within the design process.

- the document management system:
  Based on conventions for the naming of documents, the document management system assigns documents to design steps and to tools.

- the system management system:
  In this section the design method may be adapted by the user for his special requirements. Specific tools can be integrated, the names of design steps can be changed as required within the user's organisation. The installation of new projects and their assignment to designers is also included here.

- the data base:
  The documents and libraries are stored on files, which are distributed over several computers and work stations. They are made available to all users over all connected computers and work stations.

In the next chapters we will discuss the functional subsystems in more detail.

3. SUPPORT OF THE DESIGN ACTIVITIES

This is the central part of a system design environment. It gives the user the support for the method to be defined.

A method is the description of a sequence of steps that transform an initial state into a desired final state.

In our example we want to proceed from the concept for a new computer system to a design specification where all the components are identified, the so called register transfer level. Then, the concept of the computer design on a higher abstraction level must be defined.

The design of a computer means the making of a specification according to certain requirements, down to a specific degree of detail. It includes checking of the correctness of the specification as compared to the requirements.

Using a method for the design of a computer will guarantee a safe design.

The criteria for a safe design are as follows:

- the design specifications are complete and correct. This means that they are not contradictory to their requirements and the rules for a certain specification technique are fulfilled.

- the design process is reproducible. This is made possible by the existence of all the design documents, all their versions, their connections and their dependencies.

- elder design results may be reused. All the results are stored in a manner in which they can be referred to in an other design project.

- tools are used in the right place and goal-directed.

3.1. The Method

Our method now proposes a two dimensional structuring of the design process. The first dimension includes the design steps which are similar to conventional life cycles. Our steps are:

- requirement specification
- design specification
- validation and verification of the design by making a model and running the model
- presentation of the results
- design decision. This involves comparing the results to the requirements, deciding whether there must be changes in the design specification and deducing new requirements from the results ot the lower design levels.

The second dimension of structuring includes the design or abstraction levels:

- the definition of the goals for the planned system together with some general requirements
- functional system architecture: the functional parts of the system are identified. After this level a distinction is made between the hardware and the software design. Our system design environment supports the hardware design.
- the hardware architecture: all the functions that were found to be realized as hardware are distributed here and the architecture of the major components is produced.
- module architecture: the major components - for example, storage, central unit, bus and peripheral components - are structured into modules.
- register transfer level: on this level our system design environment has the connection to conventional CAD-systems.

By combining these two dimensions we can build a matrix (Fig. 2). This design matrix represents our design method.

Design steps / Abstraction Levels	Requirement Specification	Design Specification	Validation and Verification	Presentation of the Results	Design Decision
Goal Definition					
Functional System Architecture					
Hardware Architecture					
Module Architecture					
Register Transfer					

System Design Environment: Design Matrix

figure 2

Our method proposes making the design in a sequence of design steps that is given by running through the matrix from the upper left corner to the lower right one, line after line. Within the method there are feed back loops from the end of each line to its beginning. These are actually decisions that a design does not fulfill the requirements at a certain level and requires that a design must be changed and that these steps must be repeated. The user may even find that a requirement has not been achieved and thus require it to be traced back to a higher line in the design matrix. Thus all previous steps must be repeated.

3.2.Tool Support For The Method

After having established a design method we need to support the method with tools. There should be at least one tool for each square in the design matrix (except for those in the decision row). Since the tools will support the design steps they should work on all abstraction levels.

For the requirement specification we considered SADT to be an appropriate technique. A tool that supports it was integrated with some adaptations for the application in the field of hardware design.

For the design specification we choose the Specification and Description Language SDL that was developed by the CCITT (1983) for the specification of telecommunication systems. SDL provides structural concepts for a stepwise refinement technique. Thus, it can easily be used over several abstraction levels. The structure is described by a set of interconnected blocks. Nested blocks are allowed. The basic blocks contain functional descriptions such as state transition diagrams in a flow chart manner. SDL may be used in a graphical or a textual representation. The strength of the language lies in the structuring of a system and the language constructs to describe the communication between components of a system. We use a SDL tool that supports the graphical representation.

The next tools serve for the validation and verfification of a design specification. Since we believe that this design step is the central point of our system design environment we offer for it a variety of techniques and related tools for different aspects and goals. Some of these work only on specific abstraction levels, whereby others can be applied over all levels. We offer a Petrinet tool, especially for the analysis of communication conflicts. For performance analysis we have analytical and numerical tools, based on the theories of queueing networks. They can be used for the validation of complex simulation models as well. For simulation we have two simulation tools, BORIS (Siemens 1985) and SIGMUS (Kramer, Gorissen 1985), developed within our department. Both have the structure of a system as basic principle but the descriptions of the components are different. In BORIS the behaviour of a component is described in PASCAL, and nearly everything can be described. SIGMUS offers special constructs for the description of processors on the instruction set level as well as general constructs that are FORTRAN-like. We are plannig to combine some of these tools, such as to combine analytical methods with simulation.

The next row in our design matrix is the representaion of the results. We have a tool for the graphical representation of simulation results. In addition a function for statistical analysis will be added.

Only the last step cannot be supported by a tool. The evaluation of the results with the requirements must be made by the designer himself.

3.3.Connection Between The Tools

We see the tools as interrelated construction, rather than isolated entities. The result from one tool can give some input for th. next one. From a requirement specification you can deduce some basic structures for your planned system. From a SDL design specification we see the possibility of deducing a BORIS simulation model (Zorn 1984,Hogrefe and Zorn 1986). The combination of a simulation system and a tool for the representation of results is very easy. With these tool-connections we can make the design process much safer. There are fewer possiblities of errors since as much as possible is automatically done by the design system. Similar ideas were raised by Zeigler (1984) and later refined by the experimental frame concept of Rozenblit (1984,1985).

4. The Information System

The information system provides the user with information about the next design steps to be executed. The basis for this is the path through the design matrix that is proposed by the method. Proposals for the next activities are given. More general information is related to the actual abstraction level, the tools that are available and to their use. Within the information system the commands of the system design environment can be explained. Special information explains the documents that already exist for a design project. The user may also obtain the content of the libraries where the already designed components are stored for reuse in other design projects.

5. The User Interface

The user interface is the part of the system that makes the connection between the user and the tools and the commands in the system design environment. Our goal is to present a uniform interface for all tools. Since in the present version we have integrated tools which already exist, and some of them cannot be changed uniformity has not been achieved for all tools. In the optimal version the function of a tool should be presented and started by the original interface as well as from another program outside of the tool. With such tools we can achieve an optimal user interface. Within the user interface graphics will be used wherever useful. Most requirement and design specification tools are based on a graphical description. We shall try to continue with this representation in the validation step as well. Graphics are very close to the conventional way a computer designer works. In the graphical representation of the tools menues display the basic symbols to be picked up and copied for constructing design objects. The basic elements are familiar to the designer; so it won't take too much effort to learn their handling. For each tool the same functions are offered like editing, looking at the contents of an object, printing, transforming/compiling, deleting and executing. These correspond to the tool's specific functions. A browser function coupled with the document management helps the user to quickly run through connected design objects.

6. Document Management And Data Base

All the objects that are created during the design process are described by documents. Documents are referred to by their names. The names are created by means of a naming convention. Prefices allow the design step and the abstraction level to be defined uniquely. A suffix gives abbreviated information about the tool which was used to create this document. A version number is also included. By utilizing the naming conventions the other parts of the system can make an unambigous classification of every document.

The contents of a document is divided into two parts: the relation part and the description part. The relation part gives the logical connections within a design project. All documents from whose content the composition of the actual document is deduced are listed. Documents that depend upon this document are automatically inserted later. With this information the dependencies inherent to the requirements can be traced through the whole design project. These connections are the basis for checking the consistency of the documents and the browser function of the user interface. The description part carries the information about the document, it's content.

The physical storage of the documents is performed without a data base system. Documents are mapped on files. The user interface with its graphics runs on a workstation and some of the tools still run only on a mainframe computer. So the design data are spread over several computers and workstations. For such a configuration the mapping on files was the most appropriate implementation. In addition to this usual data base systems did not seem to be suited to handle the changing information structure of a design project.

7. System Managements System

This part is provided for a system manager. He installs new design projects and new users, assigns users to projects, and defines access rights to the corresponding documents. In later versions another function will be implemented that allows changes to the design matrix and the integration of other tools. Changing the matrix means that you can establish another method, with another number and naming method for both the abstraction levels and the design steps. In this way given organisations can be supported by our system design environment as well. The tools for given methods can be integrated: a relation between a new suffix for the documents and the related tool is established; furthermore the functions of the tool are bound to the commands of the user interface. The most difficult task at integration is the connection between the output of the predecessor design step and the tool's input and output and the input of the successor design step.

8. Conclusion

In this paper we presented a system design environment supporting the development of digital systems from the first requirements to the design of the register transfer level components. Instead of isolated use of all the proposed tools the system design environment provides the user with a tightly coupled set of tools. From the first decision until the last one every step within the design process may be backtracked to its requirements. In this way the design of a new system becomes much faster than it is today. Since as much information as possible is given automatically from one design step to the other there are fewer possibilities of error.

The first version of the system design environment will show the usability of the proposed tools in the context of hardware design. In later versions other tools may be added according to the user's requirements.

Acknowledgement

This work has been supported by the Federal Department for Research and Technology of the Federal Republic of Germany. The author alone is responsible for the content

References

CCITT (1983). Specification and Description Language SDL, Recommendations Z.100-Z.104, I.T.U., Geneva.

Hogrefe, D.; Zorn, S. (1986). Simulation of SDL Specified Models. In: Computer Networks And Simulation III (S. Shoemaker ed.), North Holland, Amsterdam.

Siemens(1985). BORIS-Manual, Siemens, Muenchen.

Kramer, H.; Gorissen, J. (1985). Modellierung und Simulation von Prozessorsystemen mit dem Simulationssystem SIGMUS.
Messung, Modellierung und Bewertung von Rechensystemen: GI/NTG Fachtagung. Informatik Fachberichte 110, pp. 36-50, Springer Verlag 1985.

Rozenblit, J.W. (1984). Structures for a Model-Based System Design Environment, Technical Report,
Siemens AG, West Germany.

Rozenblit, J.W. (1985). Experimental Frames for Distributed Simulation Architectures, Proc. of the 1985 Distributed Simulation Conference, San Diego, California.

Zeigler, B. P. (1984). Multifacetted Modelling and Discrete Event Simulation, Academic Press, London.

Zorn, S. (1984). Simulation of Protocols by Modelling SDL-Diagrams, Proceedings of the IASTED, Applied Informatics, Innsbruck.

Zorn, S. (1986). Computer Aided Sytem Design. In: Cybernetics And Systems 86 (R. Trappl ed.), Reidel, Dordrecht

Biography

Stephan Zorn was born in Augsburg, Germany. He studied Mathematics and Computer Science at the Technical University in Munich. He received his diploma in 1979 and joined the Corporate Research Laboratories for Information Technology of Siemens in Munich. There he started work on the design of description languages for data communication protocols and then he became the leader of several simulation projects. Presently he is the head of a group working on simulation systems, their application and the embedding of simulation into the system design process.

Stephan Zorn
Siemens AG, Corporate Laboratories for Information Technology, ZT ZTI
Otto-Hahn-Ring 6,D-8000 Munich,FRG
Tel. 011-49-89-636 44632