Decision Support, Planning, and Data Management of Complex, Discrete Event Systems

F. Bretschneider, C. Kocourek, S. Mittrach
Siemens AG, Corporate Research and Development
Otto-Hahn-Ring 6, 81739 Munich, Germany

J. Rozenblit
Department of Electrical and Computer Engineering
The University of Arizona
Tucson, AZ 85721, U.S.A

Abstract

Complex software environments, such as CAD systems, consist of many modules which are often created, maintained, and executed separately. When such systems are invoked, appropriate execution sequences and parameters must be determined for each module so that desired results are produced. The task of planning, executing, and supervising the CAD environments has become so complex that it itself requires advanced computer support facilities. This paper presents a sophisticated new design process management system (DPMS) that helps to manage globally the design process in order to achieve adequate (if not optimal) design results. It thus makes design systems more autonomous by freeing resources for the critical design steps. The architecture of the system, as well as its data management, decision support, and planning facilities are described in detail and a prototype system is presented. The system has been used in the electronic design automation area. However, the concepts described here are applicable to other design domains as well. [1]

1 Introduction

Complex design projects, such as designing systems containing hardware (HW) and software (SW) components, depend heavily on advanced computer support. In the last decade an array of computer-based design tools have been developed. Usually, such tools can be applied to only one aspect of the overall design task (e.g., routing or logic synthesis). Thus, the overall management of the design process cannot be supported. The management tasks include the monitoring of the process, ensuring the proper sequencing of the tool invocations, controlling the allocation of resources (personnel, machines, etc.), and supervising the fulfilment of design constraints. Due to frequent changes of design methods (caused by rapid technological advances), the growing object complexity, and the increasing pressure on shortening the design cycle, the task of organizing the design process has reached the limit of being manageable without the computer support. The consequences are: cost and time overruns, erroneous design results, or failures to design a system. This paper presents a step towards making design systems more autonomous by introducing a *design process management system (DPMS)*. The DPMS helps to determine appropriate sequences of design steps solving a given design problem. To this end, it determines an initial plan, estimates the consequences and design results, and then makes the necessary adaptations and modifications of design activities in close cooperation with the designer. Once a plan has been set up, it can be executed and supervised automatically. Replanning is invoked whenever the plan significantly differs from the actual state of the project.

During the plan construction, decisions must be made among alternate ways to pursue the design goal. This is done by a special decision support subsystem which uses knowledge provided by various knowledge sources (descriptions of design tools and partial processes, design experience, etc.). A data management system provides access to these knowledge sources.

The article is structured as follows: after discussing some related research in Section 2, we present the

[1] This research has been partially funded by the EC within the ESPRIT project 7364 JESSI-Common-Frame

0-8186-4020-0/93 $03.00 © 1993 IEEE

architecture of the DPMS system. In Section 4, we look at design process modelling, design flow management, decision support, data management, and planning subsystems in more detail. Our prototype system is described in Section 5. We close by looking at future research goals and possible extensions to the current prototype.

2 Related Work

Two basic approaches towards solving the design management problem are discussed in the literature: a) the use of a static process model that globally defines the dependencies among design steps. The model is created before a design process begins. It controls the sequence of steps in subsequent design projects [19]; [14], [12], [1], and b) design steps are described independently of each other by defining their functions and the conditions for activating them. These local descriptions are evaluated dynamically during the course of a design project to construct the sequence of required design tool invocations [15] [6], [9].

Procedural models or data flow graphs were used to implement the first approach. Procedural models require a fixed control flow built into the algorithm. They cannot easily express activities the order of which is determined dynamically. Data flow graphs lack the modeling power to represent complex relations among design activities, such as dynamic constraints that cannot be anticipated before earlier design stages have been completed (see Section 3). The second approach easily causes an unpredictable behavior of the overall design system. Small changes to the design step models can have unforeseen consequences for the overall process. Maintaining consistency among the models is complicated. Thus process data cannot be changed easily. Conflicts among design steps are only detected dynamically. This makes it difficult to devise conflict resolution strategies in advance. This is why more and more researchers apply mixed process models. They either differentiate between statically defined design tasks and dynamically controlled design activities [8] or do not use any a priori process descriptions at all [7]. Dynamic constraints are merely recorded during the design process. On this level, it is not possible to provide guidance for the designer, to plan or to optimize design sequences in advance. In [5], the design process is also decomposed into design tasks. Task schemas are specified as dependency graphs between abstractions of design tools and data. A design problem is first separated into tasks by the designer or a design adviser [13]. Then, the task schemas are used to derive design plans for each task. While the separation of task and process levels seems beneficial, the use of different modeling paradigms on these levels is arbitrary.

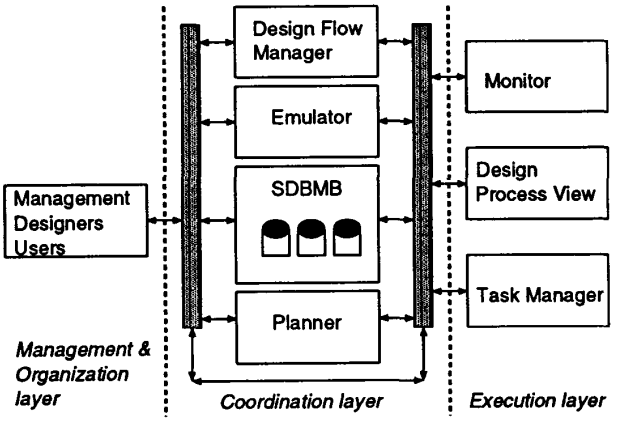

Figure 1: DPMS Architecture

3 Design Process Management System – Architecture

The formal basis and implementation of the design flow management and planning system are thoroughly described in [3]. Here, we propose a conceptual architecture of an advanced design process management system (DPMS) which complements the one already in place at Siemens AG. The DPMS architecture (Figure 1) follows the notions of Design Flow Manager, Task Manager, and the overall Petri net-based formal methodology [3]. The architecture fosters a high level view of intelligent design support. We briefly describe its components and their functionality. There are three basic layers in the system:

1. *Management and organization* layer, which determines the overall system goal (i.e., design goal) and supports interaction with the designers via the interface unit. Essentially, this layer can be used by project management personnel, corporate officers, and designers/users.
2. *Coordination* layer, which supports design task planning, decision making, diagnostics, and tool scheduling. This layer interfaces with both the users and the execution stratum.
3. *Execution* layer, which carries out design actions determined at higher levels through the Task

Manager. This layer also monitors the design process and the state of the system being designed.

These layers are modeled after architectures for high autonomy systems [18].

As indicated above, the management layer consists of corporate policy makers and system users/designers. The coordination layer is centered around a principal database called Supporting Data Bases and Methods Bank (SDBMB) (Figure 1). We envision this database as a collection of tools and procedures offering several types of design decision making. The components of this module are described in detail in Section 4. Based on this architecture, decision support is achieved through the following modules and functions:

The *Planner's* function is to generate a design process flow model and a schedule for executing it (see [3] for a good summary of planning methods). It generates a nominal plan — a pre-planned sequence of design actions that would normally be executed to realize the design goal. This process is supported by knowledge about the design goals, constraints, requirements, as well as by tools and procedures available in the SDBMB (i.e., the Tool Base, design methodology from the Methodology Base, and the design traces).

We postulate that the Planner be capable of re-planning and dynamically updating the design actions. This can be based on several factors: a) the re-plan order from the diagnosing component of the SDBMB. Re-planning would be necessary if and when the design process simulation state is significantly different from the actual process execution state, b) Inference Engines and Optimization Methods may be used for updating the plan dynamically based on locally selecting "best" design decisions using design models, and c) designers can override the plan and the *Design Flow Manager* (see below).

The *Task Manager* is responsible for invoking individual tools as directed by the *Design Flow Manager*. The Design Flow Manager has the knowledge of the nominal plan, the current state of the design process, and the state of the design object on which the process and the tools operate. As CAD tools are invoked, the states of the process and design objects are updated. Optimization and inferencing procedures should support the Design Flow Manager in its actions at two levels: reasoning about the design process and reasoning about the system being designed.

The *Monitor* "observes" the progress of both the design process flow and the design artifact. It can compare the current state with that of the expected progress based on the simulated data obtained from the *Emulator* and from the CAD simulators in the SDBMB. The Emulator executes the design process model whereas the CAD simulators operate on the design object model. The Design Flow Manager makes decisions regarding the progress of the design process and invocations of various CAD tools. It may issue a re-plan order to the planner as well.

We now proceed to discuss the details of design flow management.

4 Design Process Management Services

4.1 Process Models

We found that high level Petri net models can provide an adequate and intuitive representation of design process dependencies if properly extended for that purpose [3]. We therefore based our process modeling language on the Petri net paradigm.

High Level Petri nets [11] are directed, bipartite graphs (consisting of places and transitions) marked with tokens of various "colors". Transitions, representing the active components of a system, execute (or fire) depending on the existence of certain tokens on their input places. The firing causes the input tokens to be removed and new tokens to be created on the output places. Petri net models are compact in size and they can clearly express various relations among process steps and objects (such as sequences, conflicts, and concurrencies among activities). They define a formal behavioral semantics based on markings (i.e., the set of tokens that is present on all places) and a simple firing rule. The graphical representation of Petri nets offers an overview of the process and of its current state. The correct behavior of a Petri net (with respect to a given specification) can be tested by simulation and verified to some extent by formal methods [3], [4].

The basic Petri net paradigm is too restrictive for modeling design processes. Therefore, extentions include special decision nodes, various arc types, methods for accessing and creating sets of tokens, hierarchical transitions and places.

Like the process models themselves execution of the process models is recorded in a Petri net structure as well, and is called a trace. The transitions of a trace represent executable design steps, while the places model data object instances that are used or created by these design steps. Tokens indicate whether

the data objects are valid or not. An important property of the traces is that they are acyclic. Petri nets and traces are described in more detail in [3], [4] [7], [14] and [17].

4.2 Design Flow Management

Design Flow Manager operates on defined process models (see 4.1) to control the flow of design, i.e., to determine the current design state and the actions that can be taken next. The design flow manager supervises all contraints among design steps that are defined in the process models. Additional services that the design flow management can provide are: design process simulations, gathering of design history information, support of backtracking for the investigation of design alternatives, automatic restoration of design data and the enforcement of design policies.

4.3 Design Decision Support (DDS)

Design decisions are required in every phase of the design process, at both the process model level and the design artifact level. Here, we focus on the process model level. DDS supports users in decisions such as tool selection, design strategy and trade-off analyses. DDS also supports design planning.

In addition to the existing design flow process modeling techniques, we are introducing other design decision support concepts. They amalgamate traditional decision support systems (DSS) [20] and AI techniques.

The ultimate goal of our efforts is to design a design decision support methodology that can be easily adapted to changes in the problem domain and to varying levels of the users' decision making expertise. To this end, the core of our DPMS architecture is the SDBMB database. It consists of domain-dependent and independent knowledge sources which support the operation of the management, coordination, and execution layers of our system. The data base modules are summarized below.

The *Design Process Model Base* contains generic process models for a design domain, for instance, Petri Net models as developed by Bretschneider [3]. The *Design Object Model Base* represents models underlying the components of a design domain. Knowledge representation schemes support the organization of this database. The *CAD Tool Base* is a domain dependent library populated with CAD tools. An *Experimental Frame Base* is is intended to contain generic simulation experiment templates used in the evaluation of design models [18].

The *Design Methodology Base* and *Technology Database* are intended to store design guidelines, rules, standard design procedures for a domain, off-the shelf components and standards.

For the appropriate control of the flow and design history management, the *Design Trace Base/Trace Selector* stores traces as designs are carried out. Kocourek [16] suggests a scheme to discard traces as new or better traces are added to this database. This scheme underlies a module called a *Trace Selector*, also part of the SDBMB.

The *Optimization Methods Bank* is as a set of mathematical optimization procedures for analytical performance and tradeoff studies. The *Inference Engines* are expert system shells for the generation and selection of alternative designs.

In addition to the above modules, we envision higher level procedures such as *Design Knowledge Acquisition Methods* - methods for knowledge elicitation, used to set up the DPMS environment for a domain specific application; *Design Rationale Capture Methods Bank* - a set of procedures for design capture; and *Diagnostics Methods* - a set of procedures for design process and object fault detection and identification. We imagine this as a support tool, which can assist in recognizing that an erroneous design state (in the process and/or design object) has been produced, and identifying why it has been reached. This is a high level design support, yet to be clearly defined.

4.4 Planning

Using the information contained in the process model, the DPMS can ensure that only those tasks are activated that are applicable to the current design state. However, a combination of "correct" design steps can still lead to a situation which makes it impossible to achieve the desired design goal. Required design steps might not be enabled (due to a violation of constraints earlier in the process or missing design data), the project might exceed the available resources (e.g., time, cost), or the generated results may be useless since important design requirements are not met. To avoid these costly problems, designers and project managers need to be supported by a design planning facility. Planning services can be provided based on our process model.

The process model not only describes preconditions of design steps but it also represents their postconditions. Therefore, it can be used to simulate design processes. During simulation, instead of activating the respective design tasks when firing a transition, the output tokens of the transitions are created di-

rectly, based on the information contained in the process model. Each design process simulation leaves a trace, just as the "real" execution of the process would do. Provided that the process model is correct, the tokens that are produced as a result of a simulation run represent design data or design states that would be created if the simulated process were actually executed. The trace of the process simulation can therefore be seen as the representation of a plan for creating the design results from the state in which the simulation was started.

Using automated design process simulation, a forward-chaining planning technique has been implemented [3]. Starting from the current project state, a number of applicable design steps are selected, and their execution is simulated. From the resulting markings, one proceeds in the same direction, until the desired goal state is reached. Since only abstractions of design steps are provided in the process model, their detailed behavior has to be predicted using estimators for the activities. The challenge is to limit the effort of estimating design results, and yet to be able to determine a useful plan.

From a Petri net point of view, planning can be seen as the task of finding sequences of transition firings that, starting from an initial marking of the net, generate a new net marking containing a given set of goal tokens. One way to solve this problem is to successively enumerate all reachable markings until the marking covering the set of goal tokens has been found. This technique, which corresponds to the state-space planning approach suggested by Knapp [15], is very inefficient since the number of intermediate markings is often immense, especially if there exist numerous concurrently enabled transitions. Traces are a more efficient technique, since only the enabled transitions rather than all the intermediate markings have to be kept. Only slight restrictions to the net topology are required to guarantee that all possible firing sequences are found using this approach. The planning technique is further enhanced by applying the rules annotating the transitions in the net to select the most promising continuations.

Detail design object properties (e.g., the area of a VLSI circuit) are not usually modeled in the design process model and therefore need to be predicted by estimators for the design tools during planning. Our system allows us to declare these estimators so that they can be automatically activated and chained for providing (prospective and retrospective) measures to support planning decisions. After a design plan in the form of a trace has been determined, activity net descriptions [17] are automatically extracted. Networking techniques can then be applied to determine optimal schedules for the design activities. These identify the earliest (latest) start and completion times for the design activities, critical design steps, and optimal resource allocations.

4.5 Data and Knowledge Management

A process management system requires data management facilities. Various types of information about the system and the state of a process have to be gathered, and must be efficiently retrievable (e.g., design data, tool data, trace data, knowledge bases for decision support). All data and knowledge should be managed by a common management system. This eliminates redundancy and makes the data uniformly accessible. The data management system is the interface to data bases, file systems, or other storage mechanisms. It offers facilities for data security, data persistence, transaction mechanisms, and access control. In our case, not only design and tool data must be managed and stored, but also knowledge-based flows in the form of rules and facts, and decision data in the form of many knowledge and data bases. The use of knowledge-based tools, especially design decision support tools, requires new facilities for representation and evaluation of knowledge. Here, a solution for the handling of design knowledge represented in different forms is suggested. The result is the integration model for data and knowledge management. This model is realized by an object oriented data management interface between knowledge-based design tools and an object oriented data base management system. This interface extends existing framework data handling services. Most of the knowledge-based tools in the design area use rules or frames to represent their knowledge. They combine artificial intelligence techniques with conventional programming methods. Object oriented data models provide a good basis for procedural as well as for frame- and network-like knowledge representation mechanisms. New features for an object oriented data model are necessary to support rule based mechanisms, to extend knowledge base structuring capabilities, and to handle uncertain and incomplete knowledge [2].

A class library for all types of data with management methods provides a convenient extension of data representation and data handling services. This object oriented concept also offers facilities to structure the data by different criteria, such as design object, net hierarchy level, decision information, or planning aspect. This data management approach integrates data and

knowledge sources. It offers mechanisms to control, evaluate, represent, and to access design knowledge.

5 Prototype

The design process management prototype system has been implemented using OPS83 and the C programming languege in the UNIX environment. The system consists of two parts: the process model capture component and the design flow management subsystem.

To create a process model, users invoke a graphical editor, offering a predefined set of symbols to form a process net. A checking program automatically supervises the compliance of the process model with the extended Petri net syntax [3]. A process model can consist of any number of partial nets which are semantically joined by the checker through the identification of nodes having the same type and name. After the process model has been completely defined, the rule generator is started, which creates OPS83 production rules [10] for the process nets. The resulting rule set is linked together with the flow manager, the design decision rules ,and the planning rules to provide an executable design flow for the given process.

The design process management system is capable of controlling the correct execution of hierarchical PrT-nets, of invoking the design tools associated with the net transititons when they are fired, and of providing restoration and re-creation services for the tokens. Traces, which can be represented graphically, are collected automatically. The hierarchy of the process net is exploited to distinguish between partial traces representing net executions at different levels.

The prototype has been applied to various design processes consisting of a very large number of trace steps. Apart from a slight delay when activating and completing a design step (due to the communication overhead), there is no measurable performance decrease since the evaluation of rule sets and restoration of design process states is extremely fast. The superior performance of the system allows us to handle all graphical requests (such as redrawing of windows) elegantly through the OPS83 rules.

6 Conclusions and Future Work

In essence, all the concepts, methods, and techniques discussed here are intended to support design processes. There is no single, universally accepted architecture or framework that can be judged superior to all other systems. However, it is clear that a set of concepts is emerging to unify design activities across teams, tools, and tasks. The DPMS architecture presented here fosters such concepts. It amalgamates many of the tenets of CAD frameworks, knowledge-based design, and decision support systems. With an adequate distributed processing support, it can be placed in a concurrent design environment as well.

Initial design decision support can be provided at the design process level by incorporating facilities for design trace recording, retrieval, and selection. Preliminary work towards this objective has been undertaken by Bretschneider [3] and Kocourek [16]. To realize fully the vision charted in our architecture, a design object modeling support layer should be embedded in the system. A decision support system in the form of optimization procedures for both design process models and design objects should be added incoroporated in our system as well. Lastly, sophisticated knowledge acquisition, learning, and diagnostic modules must be developed. At each of the above three levels of refinement, the Design Flow Manager, Planner, and other components must be augmented with additional reasoning capabilites.

References

[1] K. ten Bosch, P. Bingley, and P. van der Wolf, "Design Flow Management in the NELSIS CAD Framework", *Proc. 28th Design Automation Conference*, IEEE, pp. 711-716, 1991.

[2] B. Boss, C. Danner, S. Mittrach, "Report on the Integration Model for Integrated Data and Knowledge Management Part 1", *SPI Applied Framework Research, JCF/FZI/013-03/26-Feb-93*, ESPRIT Project 7364.

[3] F. Bretschneider, *A Process Model for Design Flow Management and Planning*, VDI Verlag, Reihe 9, Nr. 157, 1993.

[4] F. Bretschneider, C. Kopf, H. Lagger, A. Hsu, E. Wei, "Knowledge Based Design Flow Management", *Proc. ICCAD*, pp. 350–353, 1990.

[5] J. Brockman and S. Director, "A Schema-Based Approach to CAD Task Management", *Proceedings of the Third IFIP WG 10.2 Workshop on Electronic Design Automation Frameworks*, Elsevier Science Publishers, 1992.

[6] M. Bushnell, "VLSI CAD Tool Integration Using the ULYSSES Environment", *Proc. 23rd Design Automation Conference*, IEEE, pp. 55-61, 1986.

[7] A. Casotto, *Automated Design Management Using Traces*, Ph.D. Dissertation, University of California, Berkeley, 1991.

[8] T. Chiueh and R. Katz, "A History Model for Managing the VLSI Design Process", *Proc. International Conference on Computer-Aided Design*, IEEE, pp. 358-361, 1990.

[9] J. Daniell, and S. Director, "An Object Oriented Approach to CAD Tool Control within a Design Framework", *Proc. 26th Design Automation Conference*, IEEE, pp. 197-202, 1989.

[10] C. Forgy, *The OPS83 User's Manual - System Version 3.0*, Production Systems Technologies Inc, 1993.

[11] G. Gernich, *Predicate / Transition Nets*, Springer Publishing Company, Lecture Notes in Computer Science 254, pp. 207–247, 1987.

[12] P. van den Hamer and M. Treffers, "A Data Flow Based Architecture for CAD Frameworks", *Proc. International Conference on Computer-Aided Design*, IEEE, pp. 350–353, 1990.

[13] M. Jacome and S. Director, "Design Process Management for CAD Frameworks", *Proceedings of the 29th Design Automation Conference*, ACM/IEEE, June 1992.

[14] A. di Janni, "A Monitor for Complex CAD Systems", *Proc. of the 23rd Design Automation Conference*, IEEE, pp. 145-151, 1986.

[15] D. Knapp and A. Parker, "A Design Utility Manager: the ADAM Planning Engine", *Proc. 23rd Design Automation Conference*, IEEE, pp. 48-54, 1986.

[16] C. Kocourek, "A Petri Net Based Design Decision Support System", *Proc. IASTED International Conference Applied Modelling and Simulation*, 1993.

[17] A. Pagnoni, *Project Engineering: Computer Oriented Planning and Operational Decision Making*, Springer Verlag, Berlin, 1990.

[18] J. W. Rozenblit, "Design for High Autonomy", *Applied Artificial Intelligence* Vol.6, pp 1-18, 1992.

[19] D. Siewiorek, D. Guise, W. Birmingham, M. Hirsch, V. Rao, G. York, "DEMETER Project: Phase 1 (1984)", *Research Report CMUCAD-84-35, SRC-CMU Center for Computer Aided Design*, Carnegie Mellon University, Pittsburgh, 1984.

[20] R. H. Sprague, Jr. and E. D. Carlson, *Building Effective Decision Support Systems*, Prentice Hall, 1982.