Generic Design Flows for Project Management in a Framework Environment *

E. Kwee-Christoph, F. Feldbusch, R. Kumar, A. Kunzmann

FZI — Computer Science Research Center
Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany

Abstract

Due to the increasing complexity of CAD systems, project managers, engineers and designers have to be supported in handling an increasing number and variety of highly specialized tools. Recent research activities follow the goal, to integrate these tools in a unified framework, which enables concurrent engineering based on a controlled execution of design activities within a common environment. Today, framework services are typically restricted to design data management and integration facilities. This paper addresses the often neglected, but very important problem of design flow management with special emphasis on project management strategies. In contrast to other approaches a generic procedure is proposed which is driven by the inherent interdependencies between project specification, design team and tools. This approach provides a flexible, adaptive and transparent design flow management system.

1 . Introduction

Innovative CAD frameworks provide many different services and utilities such as tool integration, enabling concurrency and ensuring data consistency, which are needed for the design of complex systems. One of the important facilities offered by a framework is design consultancy. This capability takes care of the needs which occur during the different phases of design process, such as the project specification, generation of possible design flows, selection of tools, execution of the tools etc., thus supporting the user across the entire design space and not just across a single design task. When frameworks are used for designing complex projects, the design consultant should also incorporate the project managerial aspects, since such designs are usually performed by a design team headed by a project manager. In this context, the automatic generation of design flows which takes various constraints into account, is indispensable. In this paper, we describe a design consultant called CADEC (CArlsruhe DEsign Consultant) which has been designed primarily for this purpose. It is worth noting here, that CADEC has been partly integrated within the well-known JESSI-Common Framework (JCF) [LiJa92].

The existing approaches for design consultants are very heterogeneous and are tuned to specific application profiles or domains as highlighted by the following summary of the state-of-the-art, within the VLSI domain. On the one hand, there are several systems which guide and support the designer to perform specific tasks of the design process, e.g. DAA for project specification [KoTh85], ADAM for layout and low-level synthesis [GrKP85], LASSIE for layout adaptation [TrDi89] and ULYSSES for layout synthesis [BuDi89]. On the other hand, some frameworks have been developed for the tool selection and activation, such as NELSIS [BoBW91] and CONDUCTOR [MiHE90], which are based on flow graphs. These tasks are also supported by HILDA [BKLH90], which however uses a modified Predicate/ Transition net. These above-mentioned frameworks provide a visual feedback of the design status. They keep track of the tasks which were executed, are currently under execution and will have to be executed.

In contrast to the existing systems, CADEC supports not only the designer, but also the project manager. In addition, an optimized use of resources is succoured by the proposed system. Two types of flows are used for representing the two important views of the design process — tasks and activities. As described later in this paper, this approach is more flexible than the flowmaps of NELSIS [BiBW92] or the process flow graphs [BaCh94], since both the temporal and data dependencies in the design process are separately visible. Furthermore, this independent generic knowledge modelling eases the maintainability and extensibility of the knowledge-based system. Moreover, depending on the project specification and its constraints, these two flow types are dynamically and automatically generated by the system.

This paper is structured as follows: section 2 describes the essential aspects for project management. The inter-relationships between these aspects are given in section 3. The described models are illustrated by an example from the VLSI design domain in section 4, which is followed by a short summary.

* This research is supported in part by the commission of EC under ESPRIT project 7364 (Jessi-Common-Frame).

2. Relevant aspects for project management

The project manager is primarily responsible for the management of the project. The major factors which influence this, from the design point of view, are project specification, design team, design activities and design tasks. The first two factors are self-explanatory, but the terms activities and tasks need clarifications. *Tasks* correspond to the abstract terminology for the chores in the design process, e.g. schematic editing, logic simulation and high-level synthesis are tasks in the VLSI domain. In order to perform these tasks appropriate design tools have to be invoked. Depending upon the selected tool parameters, a specific task can be executed. This incarnation of a tool with the conforming parameter set is defined as an *activity*. Several activities can be associated with a single tool.

In generating the design flows automatically, both mutual dependencies and constraints among the above-mentioned factors have to be considered. These inter-dependencies can be represented as a pyramid for better visualization (figure 1).

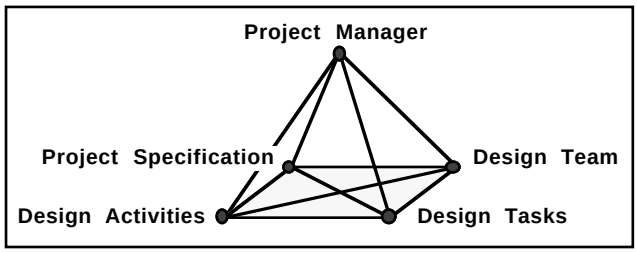

Fig. 1 : Pyramid of interdependencies

An example for the constraint interrelations is the selection of the target technology, which is part of the project specification. This typically restricts the applicable design tools and therefore the activities that have to be executed. This restriction in turn leads to a limitation of the design tasks that can be fulfilled. Additionally, the design team may also have an important impact on the selection of design tasks and activities, since depending on the availability and the experience of the team members, some tools/activities must be excluded. All these factors have to be considered in the generation of the design flows.

2.1. Scope of project manager

The duties of the project manager are to specify the project, divide it into subprojects, create an adequate design flow comprising the task flow and activity flow, schedule the chores in the subprojects, allocate a designer / design team for each subproject and to supervise the status of the overall project.

Typically the manager begins with the project specification which contains the overall constraints and the hierarchy, taking the experience of his design team into consideration. Then CADEC automatically generates the possible task and activity flows. Based on this information, the project manager can then assign specific design flows and schedules for each member of his design team. The project manager can also manually select the task flow graphs. For example, when the schedules are tight the project manager may decide that some validation is not necessary. If design quality is highly important, the project manager insists on validation tasks after every design step. Having fixed the design flows for each designer, the design process can be started. As the design progresses the system informs the project manager about every performed task, to inform him about the overall status of the project.

2.2. Project specification

CADEC supports the project manager in setting up the *requirement catalogue* for the overall project, for which specific data sheets are offered. This catalogue contains domain specific design constraints such as design style, technology, level of abstraction for starting the design in the VLSI domain. The specification also contains the hierarchical structure of the project called the *project hierarchy*, which is basically domain independent. Each subproject may be divided iteratively, thus forming a hierarchy of subprojects. The granularity of the division depends on the complexity of the specification, the available design time, the power of the available tools, etc. In the VLSI domain this division is often oriented towards basic modules which correspond to single chips or modules of a chip. Additionally, for every subproject the domain specific constraints such as the desired target technology, the abstraction level of the specification and design constraints like timing and maximum size have to be fixed. The project specification strongly biases the automatic generation of the task and activity flows as detailed in the subsections to follow.

2.3. Design activities

The availability of qualified design tools, which can be used by the designers, strongly influences the possible design methodology [RaSt92]. These tools are integrated within a framework environment, which also controls their execution. For tool encapsulation, the CAD Framework Initiative (CFI) has standardized a design tool model by describing the tool encapsulation specification [CFI-91]. In a design tool model, the relevant and significant information of each design tool must be described. One of the most important information for a design consultant system is the function of a design tool. As stated earlier, one tool can perform several tasks (= functions). In order to keep the complexity of design tool models as low as possible, it is preferable to split a tool into several activity objects for each distinct combination of input/output formats [KEHK92, KHKK93]. Therefore, this data dependency and the functions characterize the activity objects.

In CADEC, each activity object is stored a priori and can be updated during the tool integration process. Based on the project specification, an *Activity Flow Graph* (FG) can be generated dynamically from the input/output information stored in activity objects. The Activity FG is a directed graph representation of the activity flow, where the nodes correspond to activities and the edges to the input/output relationships. The edges can be used for forward and backward searching. If the 'goal' activity is specified, then backward searching is used to compose the activity flows. Otherwise, forward searching is used for a designated 'start' activity. Thus, the Activity FG shows the data dependency between the activities.

In the CADEC system so-called irrelevant activities can also be specified if these activities can always be executed automatically without any user interaction. For instance, all activities performing format conversions are of the type 'irrelevant', and the user does not have to execute them explicitly.

2.4. Design tasks

Tasks can be defined as a temporally ordered set of activities for reaching a predefined goal within the design process. This definition implies that besides the input/output data types which can be used to combine and structure the available activities, design tasks also restrict the permissible combinations of activities arising from the temporal dependencies.

Tasks represent the basic knowledge about the domain. However it is possible to take a domain independent view of the tasks, since the overall design process spans different levels of abstraction in every design domain (e.g. VLSI, Ship building, Automobiles). Within each abstraction level, there are the so-called *generic tasks*, i.e. a *specification* task, followed by the *design* and *optimization* tasks (figure 2). Each of these generic tasks can be interleaved with a generic *validation* task (V).

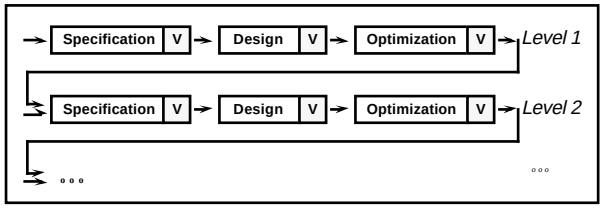

Fig. 2 : Generic Task Flow Graph

This generic Task FG can then be instantiated to a domain specific Task FG. Taking an example from the VLSI domain, the different levels of abstraction could be the high level (HL), register-transfer (RT), gate level (G) etc. At each level the generic tasks can be instantiated to domain specific tasks like high-level synthesis, logic simulation, design rule checking, etc. By using the generic Task FG representation, we can condense the information in the knowledge base for realizing our design consultant. We have incorporated rules for generic tasks, which can be instantiated to derive the temporal dependencies between the domain specific tasks.

From now on, the expression 'Task FGs' will be used to mean 'domain specific Task FGs'. However, these graphs need special nodes. This need arises from the fact that there could be a choice of domain specific tasks for performing a generic task, e.g. for the generic task design at the RT level, it is possible to have synthesis, or schematic editing as alternatives. Additionally, it is possible to execute certain tasks in parallel, e.g. placement and routing can be done in parallel with the test pattern generation and fault simulation. These above mentioned conditions lead to a choice of the 6 types of relation nodes:

- *start nodes* to represent the level at which the designer begins the design
- two types of *choice nodes*: choice begin, choice end
- two types of *parallel nodes*: parallel begin, parallel end
- *end node.*

The Task FG for the VLSI design is shown in figure 4. It has to be noted that the necessary iterations in the chores of the design process are implicitly assumed to exist between any pair of tasks. This reduces the complexity of representing the Task FGs. The relationship between Task and Activity FGs is described in section 3.

2.5. Design team

In all CAD domains, the availability of an experienced design team strongly influences the design efficiency. The project manager has to perform some kind of mapping between the design team members and the subproject specific design tasks. To achieve this, the project manager takes the designer's expertise into account and assigns tasks accordingly. In order to ease this complex mapping task, the project manager is currently supported by automatically generating the complete set of alternative flows. This procedure will be described in the following section.

3. Dynamic generation of design flows

In an object-oriented and rule-based system, the static knowledge is permanently stored in knowledge bases/ knowledge sources. In CADEC, the static knowledge is comprised of the activity objects, generic and domain-specific Task FG, and domain-independent and domain-dependent rule sets. From the activity objects and the information obtained from the project specification, the system can *dynamically* compose the flow graphs to give the users an overview about the *data* and *time dependency* of the design flow.

Figure 3 illustrates the five steps in dynamic generation of the flow graphs. Starting from the activity objects in the static knowledge base, the *Restrained Activity FG* can be generated by taking the set of available tools and the project specification into account. This in turn leads to the *Restrained Task FG*, which is used by the project manager

to make his choice in assigning a designer (design team) to a set of tasks. This results in the *Selected Task* and *Activity FGs*, which are then used by the design team for executing the overall project. The choice nodes which exist in the Restrained Task FGs, appear no more in the Selected Task FGs, since they have been eliminated by the project manager. The selected flows are subgraphs of the restrained flows.

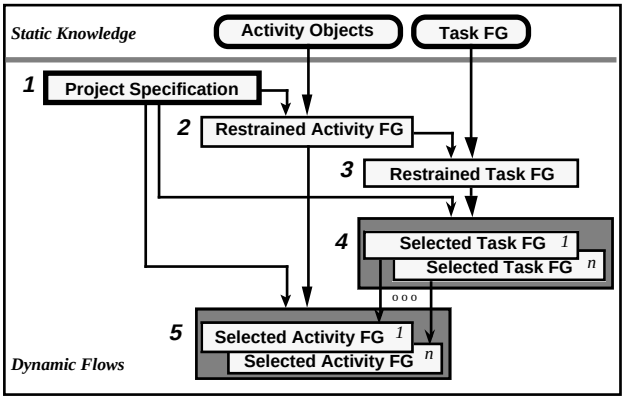

Fig. 3 : Relationship of the design flows in CADEC

The Selected Task and Activity FGs are used for controlling the execution of tools. During tool execution the designer has status information which indicates if a tool has been executed or not, thus displaying the sequence of the tools that have to be performed.

The separation of the design flows into Task and Activity FGs also eases the maintenance of the static knowledge. The number of task nodes in the static knowledge is rather constant compared to the number of activity nodes, which have to be updated more frequently to include the actual availability of tools. The rules relating to tasks and activities are also quite stable and require mostly minor modifications like the update of the input/output specifications of the activities that have been added. In the next section we exemplify the generation of the various flow graphs via an example from the VLSI domain.

4. Example

A prototype of CADEC comprising the above mentioned concepts has been implemented using the expert system shell KEE and CommonLisp. We have integrated the VLSI tool kits called SOLO, HILO and MIGRATE into this prototype, which result in 69 activity objects. Some of these activity objects have an attribute called function, which value is a task name. There exist currently 24 domain specific task objects (figure 4), which also include objects corresponding to the validation tasks such as formal verification, function verification, timing verification prelayout and postlayout. Additionally, rules corresponding to the domain dependent knowledge used in the project specification are available, including:

- *design style:* Full Custom, Standard/Macro Cell, Gate Array, FPGA, etc.
- *manufacturer:* ES2, IMS, OKI, ACTEL, XILINX, etc.
- *library:* ecpd12, gateforest, okilib, act1010, xc 2000, etc.

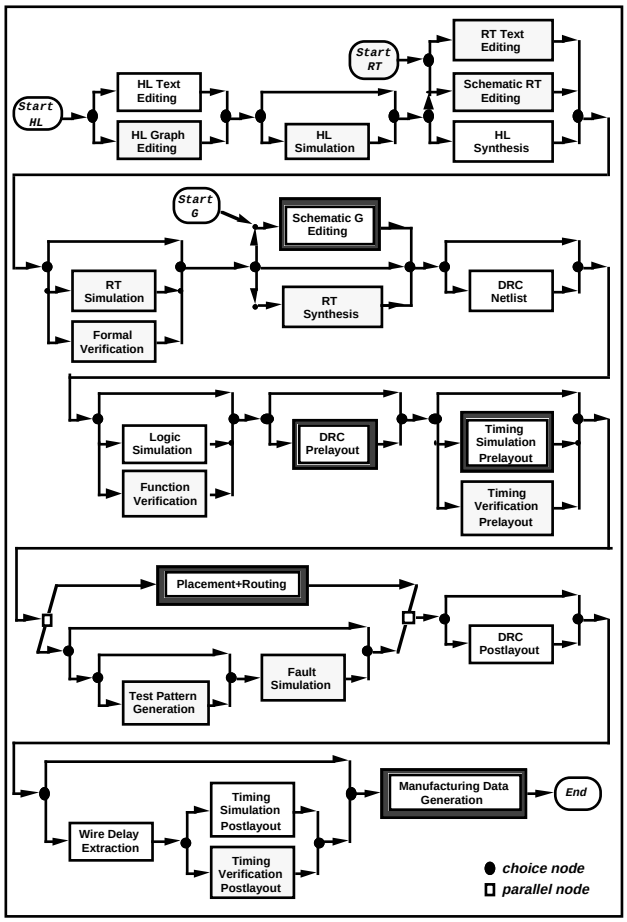

Fig. 4 : The Task Flow Graph for VLSI design

Given a constraint in the project specification which states that the design has to be performed using a XILINX FPGA, the Restrained and Selected Task/Activity FGs are generated. Based on the imposed constraints and the static knowledge, CADEC initially determines that the only appropriate tool which satisfies the constraints is XACT.
A backward and forward reasoning results in a Restrained Activity FG having 42 nodes. This Restrained Activity FG leads to a Restrained Task FG having only 12 nodes, since activities (tools) are available only for these tasks (white boxes in figure 4). The project manager can then select the tasks to be performed and assign separate flows to designers (design teams). For instance, if 5 of these tasks were selected (bold bordered boxes in figure 4), then a Selected Activity FG with 14 nodes (figure 5) will be generated. Hence, the number of task nodes and specially activity nodes is reduced. Such a Selected Activity and Selected Task FG is then used by the corresponding designer

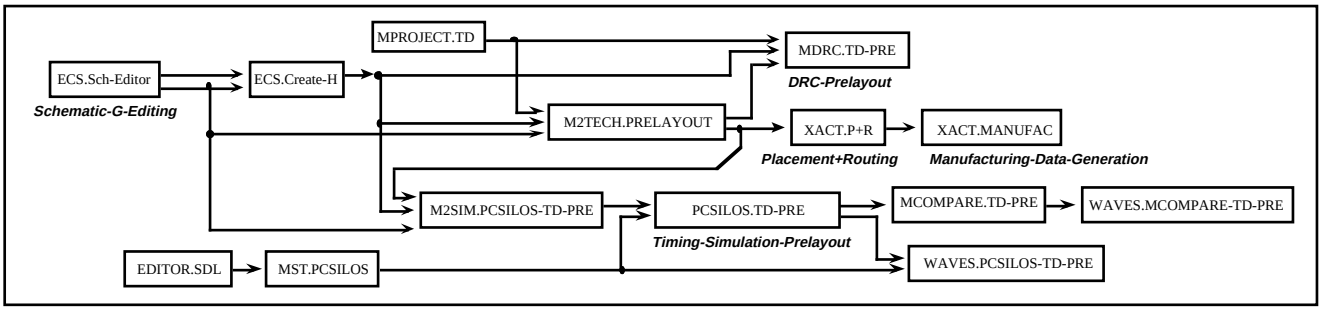

Fig. 5 : The Selected Activity Flow Graph for XILINX

to execute his set of tools. The iterations within the design which lead to cycles are handled at the meta-level [KwES93] rather than at the level of the activities or the tasks.

Some of the activity objects have an attribute called `function`. The 5 selected tasks and the 5 `functions` of the activities in the Selected Activity FG are the same, e.g. *Schematic-G-Editing* is `function` of the activity ECS.Sch-Editor. In figure 5, XACT.P+R and XACT.-MANUFAC are activities of the tool XACT, since XACT can be used for the task *Placement+Routing* or the task *Manufacturing-Data-Generation*. The tool WAVES (waveform viewer) has no attribute `function`, because it can be executed automatically (see section 2.3). Nevertheless, this tool is also split into several activities, i.e. the activities WAVES.PCSILOS-TD-PRE and WAVES.-MCOMPARE-TD-PRE, for the distinct inputs.

5. Conclusion

In this paper we have shown how design flows can be automatically generated to support the management of complex projects. Based on a large number of tools and activities that have to be executed and handled, the proposed concepts provide a comfortable generation and selection of project specific design flows. Especially, the separation of the design flow into a data dependent Activity FG and a time dependent Task FG eases this selection. As underlined by the included case-study, the CADEC prototype considers several design constraints that yield a distinct reduction of the flow complexity without losing alternative solutions.

The concept of generic and domain-specific Task FGs ensures an easy and consistent modification and extension of the knowledge base thus resulting in a very flexible design consultant for many different applications.

References

[BaCh94] Baldwin, R.; Chung, M.J.: "Design Methodology Management using Graph Grammars"; Proc. 31st DAC, 1994.

[BiBW92] Bingley, P.; ten Bosch, O.; van der Wolf, P.: "Incorporating Design Flow Management in a Framework based CAD System"; Proc. ICCAD, 1992.

[BKLH90] Bretschneider, F.; Kopf, C.; Lagger, H.; Hsu, A.; Wei, E.: "Knowledge Based Design Flow Management"; Proc. ICCAD-90, pp. 350-353, 1990.

[BoBW91] ten Bosch, K.O.; Bingley, P.; van der Wolf, P.: "Design Flow Management in the NELSIS CAD Framework"; Proc. 28th DAC, pp. 711-716, 1991.

[BuDi89] Bushnell, M.; Director, S. W.: "Automated Tool Execution in the Ulysses Design Environment"; IEEE Transactions on CAD Vol. 8, pp. 279-287, 1989

[CFI-91] Fiduk, K.: "Tool Encapsulation Specification"; in CAD Framework Initiative — Design Methodology Management, CFI-Document No. 51, 1991.

[GrKP85] Granacki, J.; Knapp, D.; Parker, A.: "The ADAM Advanced Design Automation System : Overview, Planner and Natural Language Interface"; Proc. 22nd DAC, pp. 727-730, 1985.

[KEHK92] Kwee-Christoph, E.; Eschermann, B.; Haberl, O.; Kumar, R.; Kunzmann, A.: "The CADEC VLSI Design Support Methodology"; Proc. COMP EURO-92, pp. 550-555, 1992.

[KHKK93] Kwee-Christoph, E.; Haberl, O.; Kumar, R.; Kunzmann, A.: "Knowledge Models for a Design Consultant in CAD Frameworks"; Proc. ISIC-93, pp. 391-395, 1993.

[KoTh85] Kowalski, T.J.; Thomas, D.E.: "The VLSI Design Automation Assistant: What's in a Knowledge Base"; Proc. 22nd DAC, pp. 252-258, 1985.

[KwES93] Kwee-Christoph, E.; Eschermann, B.; Schmid, D.: "A Design Consultant to Support CAD Tool Usage"; Proc. ASIC-93, pp. 301-304, 1993.

[LiJa92] Liebisch, D.C.; Jain, A.: "JESSI Common Framework Design Management - The Means to Configuration and Execution of the Design Process"; Proc. EuroDAC-92, pp. 552-557, 1992.

[MiHE90] Miyazaki, T.; Hoshino, T.; Endo, M.: "A CAD Process Scheduling Technique"; Proc. ICCAD-90, pp. 354-357, 1990.

[RaSt92] Rammig, F.J.; Steinmüller, B.: "Frameworks und Entwurfsumgebungen"; in Informatik Spektrum, Band 15, Heft 1, pp. 33-43, February 1992.

[TrDi89] Trick, M.T.; Director, S.W.: "LASSIE: Structure to Layout for Behavioral Synthesis Tools"; Proc. 26th DAC, pp. 104-109, 1989.