Flexible Workflows for Knowledge Management in the Digital Design

Mirjam Minor, Daniel Schmalen, Ralph Bergmann
University of Trier
D-54286, Trier, Germany
{minor,schmalen,bergmann}@uni-trier.de

Andreas Koldehoff
sci-worx GmbH, Garbsener Landstr. 10
D-30419, Hannover, Germany
andreas.koldehoff@sci-worx.com

Abstract

This paper presents work in progress on an adaptive workflow management tool for digital design projects. The chip design follows standardized default processes which are adapted during an ongoing project by changing requirements from both design and application uncertainties. Our approach focuses on flexible monitoring and case-based authoring support of adaptive workflows in order to support the knowledge management in a real-world application.

1 Introduction

"If EDA tools* are to assist the semiconductor industry at the 90nm† and 65nm nodes, there must be profound changes to existing tools, and the introduction of new technologies that allow designers to consider and optimize for manufacturing at each stage of the design, verification, tape-out and test process" demands Janusz Rajski, Mentor Graphics [Rajski, 2006]. The chip design in the nano era operates near the physical and technological limits – Infineon is about to develop even 45nm technology until midyear 2008. When starting projects with a new, smaller technology new types of errors may emerge. It is uncertain whether the old algorithms will work well under the new conditions. The requirements for the chip design process are high: A very tight time to market gets in conflict with the need for first time right delivery. The first layout that is manufactured has to be error free in order to avoid respins as the return to the design process and the way back to the chip foundry again are expensive and time consuming. A delay of some weeks leads to a high risk to loose the market. Like the technological imponderabilities, the uncertainty of the customer requirements may cause adaptations of the ongoing chip design process. This complex task requires a careful knowledge management.

This article presents concepts for a tool that supports the repeated reconsideration and adaptation of the ongoing design process by means of flexible workflow technology. It focuses on monitoring and authoring support capabilities for adaptive workflows. Our work is a result of the close collaboration of the University of Trier – as a subcontractor – with the microelectronics company sci-worx in the URANOS project.

Section 2 describes the digital design domain and how workflow instances evolve during the design projects by late and lean modeling. Section 3 deals with a model of context factors for authoring and monitoring purposes. In Section 4, we sketch our ideas for a monitoring that supports risk management. Finally, Section 5 contains a discussion of related work and the steps that we will do next.

*EDA = Electronic Design Automation

†nm = nanometer

2 Incremental and flexible workflow modeling

We aim to support the digital design process by a new workflow tool. It allows an incremental and flexible modeling of design processes. Initial workflow instances following a default workflow definition are adapted during the ongoing project.

2.1 Initial workflow instances

The **design flow** is a standardized description of the step by step design process for all digital design projects of a company. The initial workflow instances are derived from a workflow definition that follows that design flow. *SciWay 2.0* the design flow of sci-worx describes the four high-level phases of a project that consist of several sub-phases each:

1. Specification
2. Implementation and verification
3. FPGA synthesis and validation
4. ASIC synthesis

The specification is more than the customer requirements specification (CRS) and can take up to two month or even longer. The main work after the negotiation of the CRS with the customer is to write the specifications of the implementation, of the verification, and of the validation. The second phase, implementation and verification, includes in parallel the implementation in a Hardware Description Language (HDL), the HDL verification, and the implementation of the validation software. The synthesis is the generation of a layout that goes to the chip foundry. The verification is the functional check against the specification, while the validation (also called 'testing') is more 'hardware-related' and checks, for instance, the resistance to heat or whether a processor boots. Field Programmable Gate Arrays (FPGA's) are designed as a pre-stage of the Application Specific Integrated Circuits (ASIC's). FPGA's follow the building-block concept to be thoroughly validated before the prototype is transformed into an ASIC. Some of the testing tasks within the third and fourth phases can be automated. The literal synthesis is strongly supported by tools as well.

Sometimes, sci-worx needs to execute only a part of the design flow. Some customers come with already finished

specifications; some projects end already after the FPGA synthesis; some customers even perform the FPGA validation on their own.

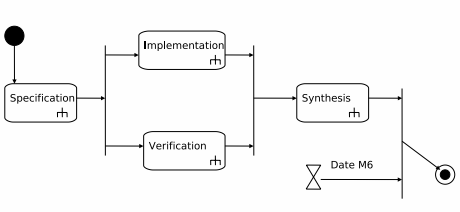

Figure 1: The workflow definition following SciWay 2.0.

Figure 1 shows a sample workflow definition in UML 2.0 notation [Ambler, 2006] following the phases and subphases of the design flow in SciWay 2.0. In terms of the workflow patterns of Aalst et al. [van der Aalst *et al.*, 2003], our workflow modelling language consists of the five basic control flow patterns sequence, parallel split, synchronization, exclusive choice, and simple merge, as well as of structured cycles (loops) and placeholder tasks for sub-workflows. The rounded boxes describe the tasks; the arrows model the sequence of tasks. Some tasks can be performed in parallel like the implementation and verification. The fork symbol in a task hides the sub-diagrams not to be confused with sub-workflows that have an own context (see Section 3). The sand-glass stands for the date of the milestone number 6 that includes the final delivery. Dates are valid for all workflow instances that belong to a project.

The initial workflow instances of a project are generated from a list of modules by means of a standard workflow definition which is more complex than the sample in Figure 1. Modules are sets of functional elements that can be tested as a unit.

2.2 Evolution of workflow instances

Modifications of the workflow instances are mainly triggered by change requests either from the customer or from the designers themselves. Besides changing a date, they may concern three types of structural modifications:

1. add or delete a task
2. split or bundle workflow instances or sub-diagrams
3. reschedule a workflow instance

Modifications within loops are valid for all future iterations. Figure 2 provides an example of a workflow instance that has been modified by an additional task 'Concretisize CRS' namely to select one of two open alternatives from 2-level or 3-level motion estimation. The reason for this modification was a change request from the designers to specify an open feature in the CRS. We decided to model an own task for it rather than a loop backwards to the specification task as the CRS document is part of the contract with the customer and can not be changed. Instead, a change request document is stored in addition to the CRS.

A workflow instance or sub-diagram can be split into several instances and sub-diagrams respectively when the implementation specification is refined or when a change request requires a finer degree of granularity. For instance, when the customer requires a simplified version of a functional element earlier than the sophisticated version, the workflow sub-diagram on this functional element is split

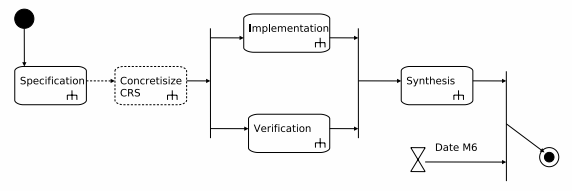

Figure 2: Extend the workflow instance on 'motion estimation' by a new task.

into two sub-diagrams with different milestones. A clone operation supports the workflow modeller in splitting both sub-diagrams or whole workflow instances. In opposite to a normal copy the clone operation skips tasks that have already been completed by the master.

A sample of rescheduling is given in Figure 3 and 4. The adaptations are triggered by a sequence of change requests from the customer. The workflow instance is on the module 'motion vector prediction'.

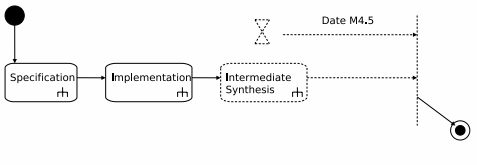

Figure 3: Remove verification and synthesis of 'motion vector prediction'.

The first change request (as depicted in Figure 3) is a result of the fact that the customer requires an accelerated schedule. An intermediate delivery M4.5 with reduced functionality has been arranged. The features will be implemented but not verified until this intermediate delivery. When the verification tasks for the module 'motion vector prediction' have to be finished is still in negotiation.

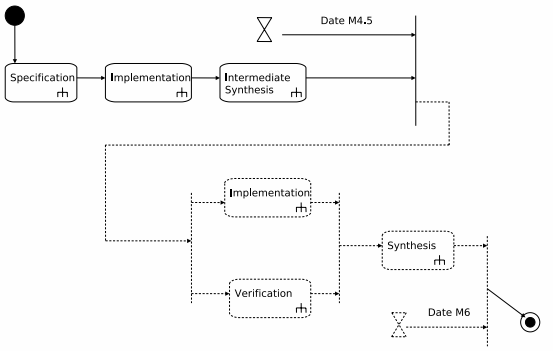

Figure 4: Reschedule verification and synthesis of 'motion vector prediction'.

Later on, a second change request is created that defines that the module shall be fully verified for the final delivery M6. That means that the workflow instance has to be extended by the verification tasks as shown in Figure 4. In parallel, there has to be done some additional implementa-

tion at least for the verification software. And afterward, the synthesis tasks follow.

2.3 Late and lean modeling

At the very beginning of a project, the functional elements of the workflow instances are only rough features from the CRS. During the implementation specification, the workflow instances are refined according to the modules of the future implementation. Later on, change requests trigger adaptations of the particular workflow instances as decribed above.

Due to this *late modeling*, the complexity of the workflow instances increases during the life cycle of a project. The granularity of the functional elements depends on the degree of agility that the project has reached. A rule of thumb is to model only things that have to be done by different persons or that deviate from the standard design flow. We do this in the roughest possible granularity and call it *lean modeling*.

The overall set of workflow instances of a project is organized within a top-level workflow instance that contains the sub-workflows for the modules.

2.4 Suspension of workflow instances

The execution of a workflow instance may take a long time. It has to be suspended during modeling activities by the user. The user may even withdraw an ongoing task from a worklist. We have developed a lock mechanism with breakpoints that are set and released by the user. Setting a breakpoint stops the ongoing tasks that lie within the focus of the breakpoint and suspends the execution of further tasks in the locked area. We have specified rules for each implemented workflow pattern how locks are propagated and released again.

A suspension may endure several weeks, e.g. if a decision of a customer is awaited. When the breakpoint has been lifted, the workflow engine determines where to continue the execution. As we do not model the data flow explicitely this is quite easy.

3 The context model

We extend the workflow model from Section 2 by a context model for authoring support and monitoring purposes. The workflow instances are embedded into a set of partially interdependent context factors. Besides factors from the design context, e.g. the employed EDA tools, we consider also the application context, e.g. the risk that the functionality might change due to the development of the end user requirements and the market. Table 1 shows some samples of context factors.

Context factors are represented as attributes with one or several attribute fields. Each attribute field has exactly one value type defined. The following value types are allowed:

- Boolean
- Single integer value
- Multiple integer value
- Percentage
- Single float value
- Multiple float value
- Date
- Time in hours
- Time in weeks
- Single value from a given pickup list
- Multiple value from a given pickup list
- Free-text
- Risk factor (has two slots: single value from a given pickup list for the *degree of damage*, *probability of occurrence* in percentage)

The value types of the sample context factors in Table 1 including the pickup lists are specified in Table 2. The underlined values are default values. For example, the algorithms risk contains an estimation how many algorithms are still risky and how important this is. An algorithm is risky when it has not yet been implemented with a certain technology. Then the performance and sometimes even the implementability is unknown in advance. The default value for this context factor is 'Cleared'. Interdependencies between units may be specified by means of binary dependencies between context factors.

The context factors are remembered for the following purposes:

- We give *authoring support* for the manual adaptation of workflow instances by retrieving similar workflow instances and presenting their subsequent adaptation steps. The case base consists of pairs of subsequent revisions of past workflow instances while the first revision of the two forms the problem part and the second, with the modification operations that have been performed, the solution part of the case. The context factors contribute to the similarity values for past workflow instances in addition to the structure and the state of the workflow instance. For example, the employed EDA tools are important to identify workflow instances with a similar context.
- Some of the context factors are *risk factors* that may be explicitly monitored by means of the system. An example is the verification grade of the specification that has to be monitored especially when external devices are employed or own IP's are reused. Furthermore, the context acquisition tool provides the utily of *risk analysis* for the current state of a project.
- Before a modification to a sub-workflow is applied the modeller may use the utility of *dependency analysis* that infers the modules (and sub-workflows) that are depending on the module either directly or indirectly.
- Some rather administrative context factors like the dates of milestones are used for *monitoring the state of a design project* (see Section 3).

Following the classification schema for workflow contexts of Maus [Maus, 2001], the context factors for the *authoring support* including the *risk analysis* and the *dependency analysis* belong to the information dimension as well as the explicitly monitored *risk factors* while the factors used for *monitoring the state of a design project* belong to the history dimension.

As for the workflow modeling, the principle of late and lean modeling holds also for the context factors: Only factors are acquired at the beginning of a project that have an important impact on the success of the whole project. Later on, they might be extended by context factors for single workflow instances in order to improve the authoring support. The values of context factors can change when the project progresses. Context factors that are not yet acquired have either empty values or pre-defined default values. We

are planning to implement a tool for the context acquisition with a configurable set of factors. The users may even wish to add context factors during runtime like specifying further milestones. New context factors are restricted to previously well-defined issues.

The monitoring of agile workflow instances is a challenging task. It allows the users to observe the state of the ongoing project and supports the risk management of crucial context factors.

The requirements for the monitoring are:

- to browse the hierarchy of the workflow instances, i.e. the module level and the level of functional elements,
- to allow the user to navigate within one instance, i.e. between tasks and sub-tasks as well as in the temporal dimension,
- to distinguish finished and open tasks, and
- to present the current state of the risk factors.

The *project managers* can use the monitoring to survey the status of the design flow including the change requests. That means the tool is able to support change request management. The annotation of context factors provides support for the risk management of projects. The *designers* can use the monitoring when joining a running project.

4 Related and future work

The CAKE system [Freßmann *et al.*, 2005] handles short term workflows and provides the dynamic assignment of sub-workflows. It uses sets of context factors to perform a case-based retrieval of appropriate sub-workflows. We are planning to extend CAKE for long-term workflows in the URANOS project.

CBRFlow [Weber *et al.*, 2005] is a conversational CBR tool for workflow management that captures the reasons for an adaptation in a dialog with the user. The workflow system ADEPT [Reichert *et al.*, 2003] allows the adaption of processes at both the process instance and the process type level. The implementations of ADEPT and CBRFlow are about to be integrated. In opposite to ADEPT, we do not model the data flow explicitely. Furthermore, the correctness of an adapted instance is not (yet) checked and our workflow definition does not yet evolve. We use the workflow definition for starting a default process that is adapted only at the instance level with both standard variations as illustrated in Figure 2 and ad-hoc changes. Potentially, some standard variations may emerge during the usage of the system and give hints for adapting also the workflow definition in future work. Like CBRFlow, URANOS will employ CBR to support the adaptation of workflow instances. URANOS will use structural CBR rather than conversational CBR, as our users are used to work with complex software tools and are supposed to be faster in specifying attribute values than in formulating questions. An authoring support agent will present previous modifications of similar workflow instances concerning the structure and the context factors of the workflow instance. This aims to decrease the manual effort for adapting workflow instances by reuse. Both CBRFlow and ADEPT do not support long-term suspension of certain areas of a workflow as it is required for the URANOS project.

FRODO task man [van Elst *et al.*, 2003] deals with late modeling of workflows, i.e. the hierarchy of sub-tasks is extensible after enacting a workflow. In URANOS, the functional split of a workflow instance is similar to FRODO's model. In contrast of FRODO, the late modeling will be supported by CBR.

The MTCT system[Bassil *et al.*, 2004] applies ADEPT for the processing of client requests for container transportation. It is related to our approach as it uses templates but in a simpler manner than we do: The templates are only for activities; the overall set of templates is fix. The focus of MTCT lies on time optimization by automatic rescheduling. In our approach, the focus lies on assisting the user who determines the scheduling by means of closed interaction with the customer.

The case handling approach[van der Aalst *et al.*, 2005] is slightly related to our work. It introduces three roles for users: the execute, the redo, and the skip role. Both redo and skip do not make structural changes of ongoing workflow instances. They are possible in our approach as well as in ADEPT, CBRFlow and FRODO.

The main issues of our future work is to implement and evaluate our authoring support concepts in a case study, and to develop a concept for the system's monitoring capabilities.

5 Acknowledgment

The authors acknowledge the Federal Ministry for Education and Science (BMBF) for funding URANOS under grant number 01M3075. We acknowledge the assistance we have received from both Stefan Pipereit, sci-worx and Marko Höpken, sci-worx as well.

References

[Ambler, 2006] Scott W. Ambler. UML 2 Activity Diagrams. Internet: http://www.agilemodeling.com/artifacts/activityDiagram.htm, 2006. [Last visited: February 2006].

[Bassil *et al.*, 2004] Sarita Bassil, Rudolf K. Keller, and Peter G. Kropf. A workflow-oriented system architecture for the management of container transportation. In Jörg Desel, Barbara Pernici, and Mathias Weske, editors, *Business Process Management: Second International Conference, BPM 2004, Potsdam, Germany, June 17-18, 2004. Proceedings*, LNCS 3080, pages 116 – 131. Springer, 2004.

[Freßmann *et al.*, 2005] Andrea Freßmann, Rainer Maximini, and Thomas Sauer. Towards collaborative agent-based knowledge support for time-critical and business-critical processes. In Klaus-Dieter Althoff, Andreas Dengel, Ralph Bergmann, Markus Nick, and Thomas Roth-Berghofer, editors, *Professional Knowledge Management: Third Biennial Conference, WM 2005*, LNAI 3782, pages 420 – 430, Kaiserslautern, Germany, April 2005. Springer-Verlag Berlin Heidelberg 2005.

[Maus, 2001] Heiko Maus. Workflow context as a means for intelligent information support. In Varol Akman, Paolo Bouquet, Richmond H. Thomason, and Roger A. Young, editors, *Modeling and Using Context, Third International and Interdisciplinary Conference, CONTEXT, 2001, Dundee, UK, July 27-30, 2001, Proceedings*, LNCS 2116, pages 261 – 274. Springer-Verlag, 2001.

[Rajski, 2006] Janusz Rajski. Shifting Perspectives on DFM, keynote talk at the International Symposium on Quality Electronic Design 2006. Internet: http://www.isqed.org/, 2006. [Last visited: January 2006].

[Reichert *et al.*, 2003] Manfred Reichert, Stefanie Rinderle, , and Peter Dadam. Adept workflow management system: Flexible support for enterprise-wide business processes (tool presentation). In W. M. P. van der Aalst et al., editor, *Proc. International Conf. on Business Process Management (BPM '03), Eindhoven, The Netherlands, June 2003*, LNCS 2678, pages 370 – 379. Springer Verlag, 2003.

[van der Aalst *et al.*, 2003] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5 – 51, 2003.

[van der Aalst *et al.*, 2005] Wil M. P. van der Aalst, Mathias Weske, and Dolf Grünbauer. Case handling: a new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129 – 162, 2005.

[van Elst *et al.*, 2003] Ludger van Elst, Felix-Robinson Aschoff, Ansgar Bernardi, Heiko Maus, and Sven Schwarz. Weakly-structured workflows for knowledge-intensive tasks: An experimental evaluation. In *12th IEEE International Workshops on Enabling Technologies (WETICE 2003), Infrastructure for Collaborative Enterprises, 9-11 June 2003, Linz, Austria*, pages 340 – 345. IEEE Computer Society, 2003.

[Weber *et al.*, 2005] Barbara Weber, Stefanie Rinderle, Werner Wild, and Manfred Reichert. CCBR-Driven Business Process Evolution. In Héctor Muñoz-Avila and Francesco Ricci, editors, *Case-Based Reasoning, Research and Development, 6th International Conference, on Case-Based Reasoning, ICCBR 2005, Chicago, IL, USA, August 23-26, 2005, Proceedings*, LNAI 3620, pages 610 – 624. Springer, 2005.

Table 1: Sample specification of context factors.

Context category	Attributes					
Risk estimation	End user and market risk			Algorithms risk		
	Technology risk			Tools risk		
Functionality	Clearness of functional specification (list of open formulations)			Uncertainty of customer requirements		
Validation	...					
Verification	Degree of verification concerning specification			Verification capability		
Technology	...					
Tools	EDA tools			Support tools		
Pins and registers	Pins/ports			Registers		
SW/HW-Codesign	Interaction software/hardware					
External devices	External IP			External software		
Design rules	...					
Milestones	M1	M2	M3	M4	M5	M6

Table 2: Sample value types for Table 1.

Attribute name	Data type	Range of values
End user and market risk	Risk	{will be fixed immediately, will cause a short loop will cause a long loop, has fatal consequences}, [0...100]
Algorithms risk	Risk	{will be fixed immediately, will cause a short loop will cause a long loop, has fatal consequences}, [0...100]
...		
EDA tools	multiple value from a pickup list	{Mentor Graphics LeonardoSpectrumTM, Mentor Graphics PrecisionTM RTL Synthesis, Synplicity, Synplify, Synplify Pro, Synopsys FPGA Compiler IITM, ...}
...		
Pins/ports	single integer value	{1, 2, ..., n}
Registers	single integer value	{1, 2, ..., n}
...		