

# Design, Integration and Validation of Heterogeneous Systems

Steffen Klupsch  
Darmstadt University of Technology  
Integrated Circuits and Systems Lab.  
Alexanderstr. 10, 64283 Darmstadt, Germany  
klupsch@iss.tu-darmstadt.de

## 1. Research Motivation

The work presented in this paper aims at improving technical products which are known as heterogeneous systems. Today many products consist of electronic parts and interfaces to the real world which include sensors and non-electrical actors. As the miniaturization continues the interrelation between the system elements is getting more important - and the system integration is getting more difficult. Therefore, integration issues must be considered during early phases of the system development. Furthermore, there is a strong need for design flows which support maintenance phases and regular product updates. The overall optimization goals are an increase of product quality and a reduction of development time.

## 2. Technical Background

System modeling is exploited to provide a link between the specification process and the implementation phase. The system model is supposed to represent an executable specification and this must be kept up-to-date during the implementation phase. The term 'system model', as used in this paper, describes a combination of models of the technical product and test environments.

An increase of quality and a reduction of development time is possible by adding system modeling to the design flow if an appropriate work flow is used. A well-adopted work flow is shown in Fig 1. Several demonstrators were built at the ICS lab to refine the work flow and to show its efficiency. Synthesizable VHDL, SPICE, VHDL-AMS, and C/C++ code were used to describe these systems which are summarized in [1]. The following sections are dedicated to presenting this work flow.

## 3. Test and Validation

The work on test cases helps to build an idea of the device needed. It often happens that a method of resolution is easier to derive based on a test szenario modeled with the testbench.

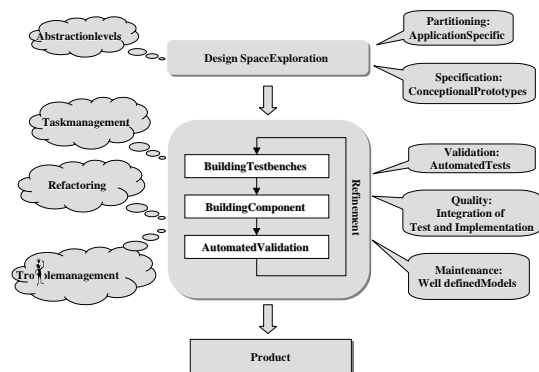


Figure 1. Draft for a quality based workflow.

The testbenches should run automatically and produce a validation result without needing user interaction. This allows to apply the testbenches on a regular basis during the development of the design. The automated validation process simplifies working on incremental enhancements because possible interdependencies are detected early.

An example for the importance of early tests is the control unit, which is part of the Dive Computer *vDC*[1]. The control unit was modeled by students who disregarded the importance of using testbenches from the beginning. They decided to describe the control unit on RT Level and built in distributed control mechanisms for accessing the RAM. During the integration phase severe problems with the memory access occurred even though the handshake protocol had been available at an early stage.

With increasing tool support for higher abstraction levels the need for technology-independent tests gains interest. The functionality realized in single entities is increasing, process technology is changing at a fast pace. System models can be used for algorithmic validation and as an executable specification. The idea of an executable specification in form of system models shows its potential if reuse is considered and ensures that the product specification is updated as it changes. The Multi Nature demonstrator *vDC*, as well as purely digital demonstrators outlined in [1], were successfully developed with system model based validation.

## 4. Task Management

Decomposition of large chip projects in smaller well-defined subtasks is a design step whose importance is well-known throughout the design flow. It is strongly based on heuristics and requires constant refinement of specifications and implementation decisions. But even if the hierarchy and specification for the subtasks is fixed, there is need for prototyping.

Prototyping is a good way to reduce the development time: prototype architectures can be used to realize key features of the algorithm - these simplified prototypes are easier to test due to reduced synthesis time and reduced complexity. This concept was used to accelerate the creation of the floating point unit *a*FPU[1]. This component implements addition, subtraction, multiplication, division, and exponentiation operations. The simplest way to select features is to write prototypes for each operation. Actually, the simplifications chosen in the example were based on an analysis of the operations. For example, the subtraction can be transformed into an addition, hence these operations are merged. The results are prototypes which can be validated within a few minutes. All prototypes should share a common entity declaration to allow the reuse of the testbenches during the integration phase.

In general, it is best to keep the design as simple as possible, but there are exceptions. The *a*FPU design uses parameters and attributes extensively to allow for arbitrary precision floating point numbers. This permits preliminary functional testing of synthesis results with reduced precision floating point numbers. The test is much faster due to less simulation elements and it is an efficient validation: algorithmic errors in exception handling and rounding issues do not depend on the bit length of the floating point numbers.

## 5. Refactoring

Refactoring is a buzzword from the domain of software engineering. The development of software projects bears analogies to digital circuit design: the project is specified, divided into subtasks which are coded, and finally integrated. If the project is successful, maintenance activities will follow and additional functionality might be added. Even though team members will change, the project's specification and the realized methods of resolution have to be saved from decay.

We apply refactoring to our hardware designs regularly. This cannot be done with the same methods software engineers use, but refactoring is applied with the same intention.

One goal is to reduce the amount of replicated code. Another goal is to simplify the models by grouping declarations and behavioral descriptions into appropriate packages,

as well as to group signals into suitable signal vectors. The *v*DC was heavily refactored. Various designers worked on the entities and each designer had his own coding style. From time to time naming conventions had to be revised and descriptions had to be simplified, partially to trace bugs, but mostly to make the design easier to understand.

Libraries should be split based on the size of the library and on the usage of library features in the design. Features which are used rarely should be moved into an extra library.

The difference between refactoring and intuitive code cleanup is the incremental approach: After each change the result is validated by automated testbenches to reduce the risk of inserting bugs. The different aspects of refactoring are taken care of sequentially. This enhances efficiency because it keeps the tasks simple. The regular refactoring of the design helps to maintain a high quality design to which enhancements can be added painlessly.

## 6. Troubleshooting

Regular discussions between the team members involved in the *v*DC project have shown that it is much easier for a non-involved observer to find bugs than for the author of the description. It is not necessary to choose an observer with better modeling knowledge, but it is helpful to do this work in groups. This will allow the observer to become acquainted with the design much faster and to find fragments where coding is strange. The author is needed to enhance his model without introducing unwanted side effects. Additionally, working in groups improves information exchange regarding modeling style, tool usage, and pitfalls of the synthesis and simulation tools.

## 7. Description of Results

The workflow (Fig. 1) was applied during several design studies to study typical problems of heterogeneous systems.

Generalized concepts were presented which enhance the workflow. Future work will focus on the further development of generic quality indicators based on the introduced working tasks [2] and on improving our system simulation environment by integrating FPGA based hardware acceleration in the multi nature software simulation system.

## References

- [1] *ICS Lab Homepage / Research / St. Klupsch*, <http://www.vlsi.informatik.tu-darmstadt.de/staff/klupsch/>
- [2] St. Klupsch, S. A. Huss, "Optimierung der Modellbildung unter Berücksichtigung unterschiedlicher Entwurfskriterien", *GMM-ITG-GI Workshop Multi-Nature Systems*, Jena, 1999, pp. 63-72.