# Manageable Dynamic Reconfiguration with EVE – Extendable VHDL Editor

*Christopher Pohl, Ralf Fuest, Mario Porrmann*

*{pohl,rfuest,porrmann}@hni.upb.de*

*System and Circuit Technology Group - University of Paderborn*

*http://wwwhni.uni-paderborn.de/sct/*

## Abstract

*Dynamic reconfiguration of FPGAs is a promising approach for saving resources, thus becoming attractive for industrial applications. In this paper we present a complete tool flow which enables users to create dynamically reconfigurable systems without in depth knowledge of the underlying hardware and methodologies.*

## 1. Introduction

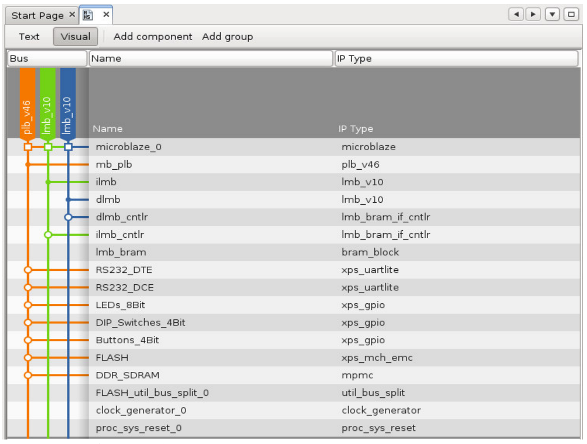
Dynamic reconfiguration of FPGAs is a way to introduce a flexible resource management into the FPGA design flow: certain areas of the FPGA are configured at runtime, such that hardware components use resources only when they are truly required. In this way, e.g.; certain tasks can be moved from software to hardware at runtime, allowing for dynamically adopting the system performance according to changing requirements. While the theoretical approaches for partial dynamic reconfiguration are becoming more and more mature, the practical use of this promising technique is limited by the lack of complete and usable tool chains. We present an approach where the user is guided through the decision making process and the generation of valid configurations by a graphical representation of the design and the reconfiguration options. Based on an analysis of the design in question, the partitioning into static and dynamic parts is done graphically by the user, and the impact on FPGA resources is estimated. Once a satisfactory partitioning has been determined, the automatic integration of communication infrastructures supporting partial dynamic reconfiguration is performed, and the synthesis flow can be started.

In this paper we give a short introduction into the architecture of EVE and into the tools involved.

## 2. Design flow using EVE

Eve is a graphical frontend for our partial dynamic design flow INDRA [1], integrated into the NETBEANS platform [2]. The input to any EVE project comprises of design files, either Xiinx EDK projects or VHDL code. These have to be parsed by EVE in order to generate a graphical representation of the design, therefore two of the main components are a parser for EDK designs and a VHDL parser (vMAGIC [3]). Both parsers have been built using ANTLR [4], and both were augmented with functionality for editing and writing their respective formats. The EDK parser so far considers the EDK project file (.MHS) and the associated core description files (.MPD) in order to generate a correct bus hierarchy. This bus hierarchy is then combined with the additional VHDL files provided by the user to form a complete representation of the design. (see figure 1).

In the next step, the user introduces groups of design components based on this graphical representation (see figure 2). These groups can then be marked as static or dynamic: static components are those, which remain untouched in the process of reconfiguration (typically a processor and peripherals controlling the reconfiguration), dynamic components are, e.g., hardware accelerators for tasks like cryptography, video processing or controllers in mechatronic systems.



| Name | IP Type |
|---|---|
| microblaze_0 | microblaze |
| mb_plb | plb_v46 |
| ilmb | lmb_v10 |
| dlmb | lmb_v10 |
| dlmb_cntlr | lmb_bram_if_cntlr |
| ilmb_cntlr | lmb_bram_if_cntlr |
| lmb_bram | bram_block |
| RS232_DTE | xps_uartlite |
| RS232_DCE | xps_uartlite |
| LEDs_8Bit | xps_gpio |
| DIP_Switches_4Bit | xps_gpio |
| Buttons_4Bit | xps_gpio |
| FLASH | xps_mch_emc |
| DDR_SDRAM | mpmc |
| FLASH_util_bus_split_0 | util_bus_split |
| clock_generator_0 | clock_generator |
| proc_sys_reset_0 | proc_sys_reset |

**Figure 1** EVE MHS Editor before grouping

An optional part in this step is the definition and integration of debug and test components into the static part. EVE supports offline (HiLDE [5]) and online (HiLDEGART) monitoring, and it provides a means to visualize the reconfiguration process using SiLLis.based component. SiLLis (Simple Language for Listeners) is a language for describing protocols, with the ability to automatically create synthesizable hardware components to monitor communication lines unsing these protocols. To integrate these test components, the designer needs to specify, e.g., number formats, sample rates, protocols (SiLLis) and the actual test points, the concrete implementation of the debug components is performed by several vMAGIC based tools.

After the partitions and the test components have been specified, the resource requirements of both the static and the dynamic components are estimated using Xilinx ISE. This step is necessary to ensure that the user-defined partitioning is feasible using the chosen FPGA. If so, it is necessary to generate and integrate a communication infrastructure for the dynamically reconfigurable areas. The infrastructure itself comprises of so called bus macros, which can be created using tools from the INDRA flow; the integration of these macros into the VHDL files, however,

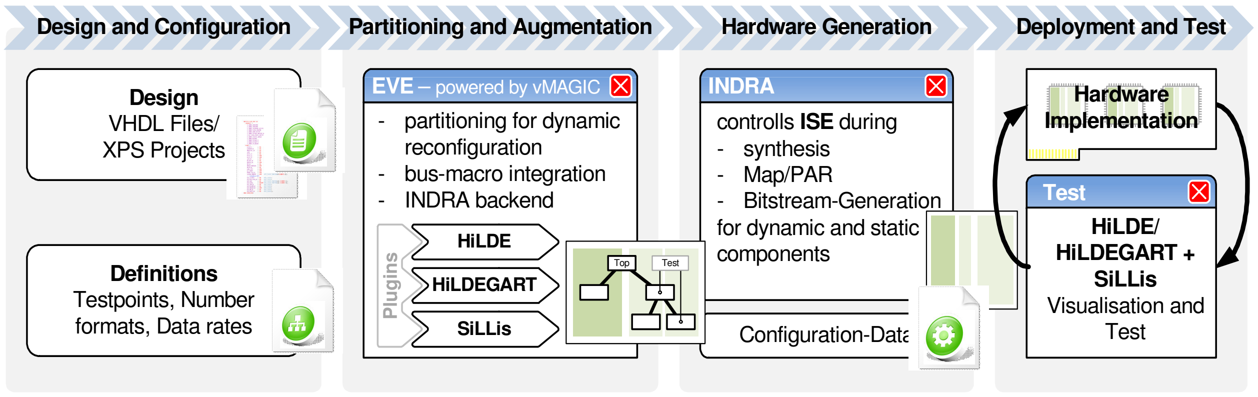


**Figure 2:** Design-flow for partial dynamic reconfiguration using EVE

is performed using vMAGIC, thus completing the design of the reconfigurable system.
After having partitioned the design, which originally comprised of VHDL and EDK components, the actual implementation (synthesis, map, PAR, bitstream generation) is carried out by the INDRA flow, while all required actions are supervised by EVE. After a successful implementation, an initial bitstream, containing the static components and dynamic dummy components, as well as the dynamic components are available. If debug and test components have been integrated, a number of tools can then provide assistance for starting the complete dynamically reconfigurable systems.
The main benefit of the presented design flow is that a lot of in depth knowledge of the Xilinx ISE tools is abstracted into a convenient user interface, which supports all steps from the partitioning to the implementation of a reconfigurable system.
An additional feature of EVE is a VHDL editor featuring syntax highlighting, template based code completion, code folding, and, because of vMAGIC, a number of features to speed up coding of VHDL designs.

## 3. vMAGIC

Our approach is based on vMAGIC, the *VHDL Manipulation and Generation Interface,* a Java library for analyzing and creating VHDL code. vMAGIC was developed by the authors to provide the functionality required in this demonstrator:

- Parsing VHDL'93 compliant code: processing VHDL code is much easier in a parser tree (AST) than in a textual representation.
- Easily manipulating code: Java classes representing design objects (signals, multiplexers, registers) have been developed to provide a structured means for manipulating VHDL code. These objects can be generated from scratch and added to existing code and vice versa, existing objects can be altered or used to generate new objects (e.g., generate an instance from an entity)
- Writing code: the newly generated code can be emitted as readable and well structured VHDL code.

For EVE, vMAGIC provides facilities to extract the hierarchies from VHDL files, alter these hierarchies to meet the new requirements (dynamic reconfiguration), and to include the communication structures (bus macros) into the VHDL design.

## 4. Conclusion

In this paper we have introduced a complete design flow for partial reconfiguration of FPGAs (EVE), where the partitioning of the system is supported by a graphical environment. In particular, all necessary implementation steps, including the generation of the communication infrastructure are started and supervised by EVE, such that the designer does not need to have an in depth knowledge of the underlying architecture.

## 5. References

[1] J. Hagemeyer, B. Kettelhoit, M. Koester and M. Porrmann. *INDRA – Integrated Design Flow for Reconfigurable Architectures.* In Proc. of the Int. Conference on Design, Automation and Test in Europe (DATE), IEEE Computer Society, 2007.
[2] T. Boudreau, J. Glick and V. Spurlin. *NetBeans: the definitive guide.* O'Reilly Associates, Inc. Sebastopol, CA, USA, 2002.
[3] C. Pohl, C. Paiz and M. Porrmann. *vMAGIC - VHDL Manipulation and Automation for Reliable System Development.* In Proceedings of the 3rd International Workshop on Reconfigurable Computing Education, April 2008.
[4] T.J. Parr and R.W. Quong. *ANTLR: A predicated-LL (k) parser generator.* Software: Practice and Experience, vol. 25(7), pages 789--810, 1995.
[5] C. Paiz, C. Pohl and M. Porrmann. *Reconfigurable Hardware in-the-Loop Simulations for Digital Control Design.* In 3rd International Conference on Informatics in Control, Automation and Robotics (ICINCO), pages 39-46, Setubal, Portugal, August 2006.

## Acknowledgement

This work was developed in the course of the "Collaborative Research Center 614-Self-Optimising - Concepts and Structures - in Mechanical Engineering" - University of Paderborn, and was published on its behalf and funded by the Deutsche Forschungsgemeinschaft.