INDRA – Integrated Design Flow for Reconfigurable Architectures

Jens Hagemeyer, Boris Kettelhoit, Markus Koester, Mario Porrmann

{jenze, boris, koester, porrmann}@hni.uni-paderborn.de

University of Paderborn – Germany

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

Abstract

INDRA offers a solution for an automated design flow of heterogeneous partially reconfigurable systems utilizing advanced features such as a flexible placement and communication for hardware modules.

1. Introduction

Partial run-time reconfiguration is a promising concept to enhance the performance of a given FPGA. By using partial run-time reconfiguration dynamic system components can be added or removed according to the current needs of the system. Dynamic system components are typically represented by hardware modules. To enable the concept of partial run-time reconfiguration a suitable communication infrastructure is needed that allows for interconnecting the currently placed hardware modules. According to the system requirements various communication infrastructures have been proposed [6, 8].

Simple dynamically reconfigurable systems consist of only a few, separated partially reconfigurable regions (typically one or two) that are connected by simple communications channels (so called bus macros). The areas of these modules correspond to the whole partially reconfigurable region. Thus, these systems have a low flexibility due to the limited number of parallelly executable modules. To increase the flexibility, advanced dynamically reconfigurable systems have been proposed [3] where the partially reconfigurable region is partitioned into multiple individually reconfigurable tiles. The tiles are connected by a homogeneous communication infrastructure. A module is composed of a group of contiguous tiles. Placing a module is equivalent to searching an area with as much contiguous free tiles as needed by the module. While each module is based on only one bitstream, it can be placed at various feasible positions. Such a system typically allows the placement of a reconfigurable module somewhere in a reconfigurable area (so called free module placement [3]), which greatly improves the system flexibility, but also introduces the need to find a suitable module position at run-time.

The design of such a system poses many challenges: For a given application a suitable device has to be selected. The reconfigurable area needs to be partitioned into a base region for static components (e. g., configuration manager, system bus arbiter) and into a dynamic region for partially reconfiguring the modules at run-time. In particular, the heterogeneity of the reconfigurable area, which is caused by distributed BRAM and DSP elements, needs to be considered. At run-time, placement algorithms with support for heterogeneous devices are needed [5]. In order to avoid introducing additional heterogeneity, a homogeneous communication infrastructure is required, which enables communication between the partial reconfiguration modules and the base region. Initial bitstreams for the base region and the communication infrastructure as well as bitstreams for the partial reconfiguration modules need to be generated. The currently available design tools do not allow a fully integrated design of a complete partially reconfigurable system, but still require an extensive knowledge in FPGA design methodology.

2. INDRA

INDRA is an Integrated Design Flow for Reconfigurable Architectures. Figure 1 shows an overview of the tools, that are integrated in INDRA. At first, the application has to be partitioned into static components and dynamic components. The partitioning depends on the properties of the selected device, such as the granularity of reconfiguration (length of a configuration frame), and on the selected placement approach. E. g., the reconfigurability of the Virtex-4/5 devices allows a two-dimensional placement of hardware modules at a granularity of a configuration frame. To enable a two-dimensional placement with Virtex-2 FPGAs, special reconfiguration methods as presented, e. g., in [9] must be used.

Synthesis of the base region and of the modules is performed based on the system partitioning. Depending on the size of the module, which is obtained from synthesis estimation, and on the inherent heterogeneity of the FPGA, INDRA automatically determines the steps that are required for the synthesis of the modules. If an example schedule of the target application is given, the run-time behavior of the system can be tested by means of the simulation framework SARA [3]. This allows testing of various placement algorithms that are used for finding suitable positions of the hardware modules at run-time. Based on the simulation results the best suited placement algorithm is selected. Additionally, the simulation can be used to optimize the architectural partitioning and the communication infrastructure.

The communication infrastructure for a dynamically reconfigurable system, which supports flexible module placement and module relocation at run-time, requires to be homogeneous. Homogeneity means, that the individually reconfigurable tiles are connected by the same routing resources. Thus, modules cannot only be placed at one dedicated position, but at any position with sufficient free contiguous tiles. To generate a homogeneous communication infrastructure, X-CMG [1] can be used. X-CMG features a

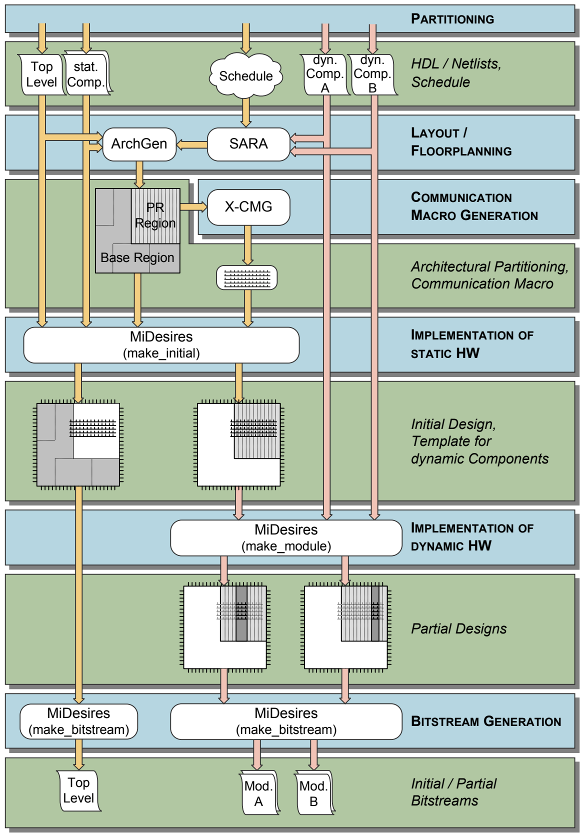
Partitioning
Top Level
stat. Comp.
Schedule
dyn. Comp. A
dyn. Comp. B
HDL / Netlists, Schedule
ArchGen
SARA
Layout / Floorplanning
PR Region
Base Region
X-CMG
Communication Macro Generation
Architectural Partitioning, Communication Macro
MiDesires (make_initial)
Implementation of static HW
Initial Design, Template for dynamic Components
MiDesires (make_module)
Implementation of dynamic HW
Partial Designs
MiDesires (make_bitstream)
MiDesires (make_bitstream)
Bitstream Generation
Top Level
Mod. A
Mod. B
Initial / Partial Bitstreams

Figure 1 : Overview of INDRA

multilevel, primitive-based communication macro generation approach. To build a homogeneous communication infrastructure, X-CMG uses primitives for different classes of signals. The primitives can be used to implement shared signals and dedicated signals as discussed in [2].
All previously described steps are part of the INDRA front-end. The back-end of INDRA is realized by the MiDesires (Module implementation design flow for reconfigurable systems) suite. It is based on the current Xilinx JTRS flow as presented in [7]. MiDesires is not only an interface to the Xilinx tools, as it provides additional features like saving and loading the current state of the design flow. Additionally, it automates required steps during the module synthesis such as the adaptation of UCF files and resolving of dependencies. Furthermore, it directly changes parts of the Xilinx flow by modifying internal PAR settings accordingly.
For evaluation of the proposed design flow and for demonstration of the resulting functionality the modular rapid prototyping system RAPTOR2000 [4] is used. The RAPTOR2000 system (Figure 2) and its successors are designed to support partial run-time reconfiguration. Up to six daughterboards utilizing state-of-the-art FPGAs are interconnected using a flexible high-bandwidth communication infrastructure. A tight coupling to the host system is provided by integrated PCI, PCI-X, or PCI-Express interfaces.

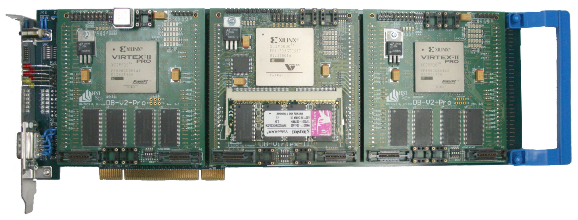

Figure 2 Self developed RAPTOR2000 Board supporting partial run-time reconfiguration

3. References

[1] J. Hagemeyer, B. Kettelhoit, M. Koester and M. Porrmann, “Design of Homogeneous Communication Infrastructures for Partially Reconfigurable FPGAs.”, *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms* (ERSA '07), June 2007. (submitted for publication)
[2] J. Hagemeyer, B. Kettelhoit and M. Porrmann, “Dedicated Module Access in Dynamically Reconfigurable Systems.”, *Proceedings of the 19th International Parallel and Distributed Processing Symposium* (IPDPS). IEEE Computer Society, 2005.
[3] H. Kalte, B. Kettelhoit, M. Koester, M. Porrmann and U. Rückert, “A System Approach for Partially Reconfigurable Architectures.”, *International Journal of Embedded Systems* (IJES), Inderscience Publisher, 1(3/4): pp. 274-290, 2005.
[4] H. Kalte, M. Porrmann and U. Rückert, “A Prototyping Platform for Dynamically Reconfigurable System on Chip Designs.”, *Proceedings of the IEEE Workshop Heterogeneous reconfigurable Systems on Chip* (SoC), August 2002
[5] M. Koester, H. Kalte and M. Porrmann, “Task placement for heterogeneous reconfigurable architectures.”, *Proceedings of the IEEE 2005 Conference on Field-Programmable Technology* (FPT’05), pp. 43-50, IEEE Computer Society, 2006.
[6] S. Koh and O. Diessel, “Comma: A communications methodology for dynamic module-based reconfiguration of FPGAs.”, *ARCS Workshops*, 2006.
[7] P. Lysaght, B. Blodget, J. Mason, J. Young and B. Bridgford, “Enhanced Architectures, Design Methodologies and CAD Tools for dynamic reconfiguration of XILINX FPGAs.”, *16th International Conference on Field Programmable Logic and Applications* (FPL2006), pp. 12-17, August 2006.
[8] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde and R. Lauwereins, “Interconnection networks enable finegrain dynamic multi-tasking on FPGAs.”, *Proceedings of the 12th International Conference on Field-Programmable Logic and Applications* (FPL05), 2005.
[9] P. Sedcole, B. Blodget, J. Anderson, P. Lysaght and T. Becker, “Modular partial reconfiguration in Virtex FPGAs.”, *Proceedings of the 15th International Conference on Field Programmable Logic and Applications* (FPL05), Finland, 2005.