DESIGN TOOLS FOR SELF-AWARE SYSTEMS ON FPGAS

Dirk Koch, Christian Beckhoff, Alexander Wold, and Jim Torresen

Department of Informatics,
University of Oslo (Norway)
email: {koch, jimtoer}@ifi.uio.no

ABSTRACT

To fully exploit the capabilities of run-time reconfigurable FPGAs in self-aware systems, design tools are required that exceed the capabilities of present vendor design tools. Such tools must allow the implementation of scaleable reconfigurable systems with various different partial modules that might be loaded to different positions of the device at run-time. This comprises several complex tasks, including floorplanning, communication architecture synthesis, physical constraints generation, and the physical implementation all the way down to the final bitstream generation. In this paper, we present how our GOAHEAD framework helps in implementing self-aware systems with a minimum of user interaction.

1. INTRODUCTION

Partial reconfiguration of FPGAs is a key technology in the implementation of self-aware systems that are capable of adapting behavior and structure of hardware at run-time. For example, reconfigurable modules might be relocated to compensate for device defects or a variable number of accelerator modules might be instantiated in order to adapt to varying compute demands. In general, such self-adaptations require that various modules (each with different resource requirements) can be placed freely and multiple times on the fabric while being able to communicate with the dynamically placed modules.

However, from the FPGA vendor side, there is only weak support for implementing such flexible self-adaptive systems. For example, following the latest partial design flow from Xilinx [1] still does not permit relocation of modules on the FPGA fabric. This means that in a scenario with, for example, 10 possible module placement positions and 5 different modules, it requires 50 individual place & route steps for the modules and consequently 50 partial configuration bitstreams. Moreover, all these physical implementation steps have to be carried out again after changes in the static part of the system. Note that these restrictions also apply for the PR tools from Altera [2].

A further drawback of the vendor tools is that they do not permit sharing a reconfigurable region by multiple modules at the same time and only one module can be placed exclusively into a reconfigurable region. For instance, a large module cannot be replaced by multiple smaller ones. Consequently, the vendor tools neither scale with the complexity required for implementing advanced self-adaptive systems nor do they allow for the implementation of systems that exploit the full flexibility available in an FPGA.

Besides the vendor tools from Xilinx, there exist a few academic approaches to implementing reconfigurable systems on FPGAs. For example, OpenPR [3] allows for the implementation of relocatable modules resulting in a more scalable flow than what is available from the FPGA vendors. However, having multiple modules in a reconfigurable region or the crossing of static routing through a reconfigurable regions is not supported. The tool ReCoBus-Builder [4] includes synthesis capabilities of communication architectures required to integrate multiple modules simultaneously in a reconfigurable region, but the tool only supports older devices. The following sections introduce how our new tool GOAHEAD [5], can be used for building self-aware systems.

2. IMPLEMENTING SELF-AWARE SYSTEMS WITH GOAHEAD

Implementing self-aware systems using partial reconfiguration on FPGAs involves several complex tasks and deep knowledge about self-aware strategies, as well as knowledge on how to build and manage reconfigurable systems. The tool GOAHEAD [5] automates and assists in the latter issues. GOAHEAD provides the following features:

- *Floorplanning* in manual and automatic mode. This is the process of defining reconfigurable regions on an FPGA for hosting dynamically loadable modules.
- *Communication architecture synthesis* for island style and slot-based reconfigurable systems. This is the process of binding signals for the communication with the partial modules to physical wires on the fabric.
- *Physical constraints generation* for place and route.
- *Design rule checking and verification.* GOAHEAD can create netlists for simulation, timing verification,

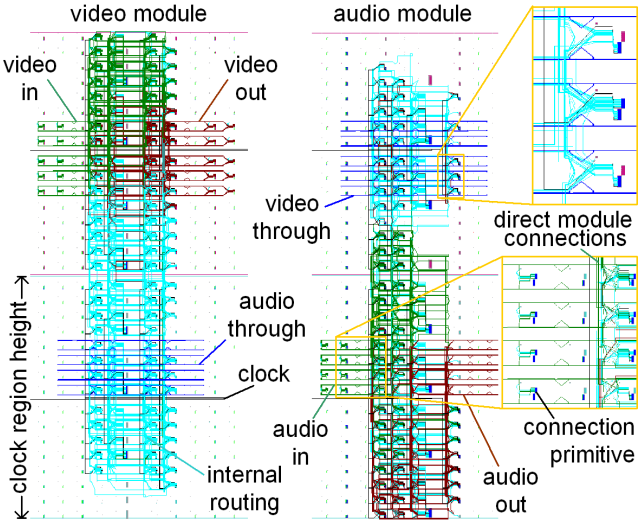

Fig. 1. Audio and video module with stitchable interfaces.

and full bitstream generation of any combination of modules that might occur during system operation.

- *Bitstream assembly* of the static and all partial module configuration bitstreams.

The communication architecture synthesis can create module interfaces that allow stitching various modules together in an arbitrary manner. It is also possible to route signals through the region of a partial module (e.g., for crossing the reconfigurable region), while still being able to relocate modules to different positions on the FPGA (as long as the resource footprint matches). See [5] for more details on module relocation.

Figure 1 shows an example of two stitchable modules. While the left module accesses the video stream while routing through the audio stream, respectively, the right module accesses the audio stream and routes through the video stream. Both modules provide connections that permit direct connections between adjacently placed modules. However, a system might provide *route through modules* in order to bridge a gap between two placed modules. The compatibility of the interfaces is ensured by constraining signals to matching wires on both sides of the module.

The order of stitching the modules is reversible, as long as the underlying resource footprint matches. Because the modules work on different data (i.e., audio and video data), it is possible to stitch the audio module either left or right beside the video module while still providing exactly the same functionality. In the case of multiple modules working on the same data stream, two modes are supported: 1) in *multicast mode* the input stream is tapped and sent directly to the next module which permits an arbitrary placement order of modules along a stream, while 2) in *read-modify-write mode* the input data is processed and the result is streamed to adjacent modules. In the later case, the data dependency results in an placement order along the data stream that has to be followed.

For high performance, the communication can be pipelined and clock rates of more than 300 MHz are possible on Xilinx Virtex-6 FPGAs. Note that GOAHEAD can include pipeline registers into route through channels that permit, for example, to partially load a reconfigurable video module without interfering the audio stream or vice versa. This permits stitching together a large number of modules without dropping the throughput on the channels.

As Opposed to the partial design flows from the FPGA vendors Xilinx and Altera, GOAHEAD does not need connection primitives on the signal paths from or to a partial module (called proxy logic [1] by Xilinx). This removes in particular for small modules (e.g., instruction set extensions for CPUs) the logic overhead and the additional latency of the connection primitives. As shown in the right zoomed box in Figure 1, input signals are routed directly to the module. In the GOAHEAD design flow, connection primitives are placed temporarily *outside* the module. However, by cutting out the module, when generating the partial configuration bitsream, the connection primitive is completely removed from the system.

3. CONCLUSIONS

Our novel tool GOAHEAD provides distinguished features for implementing reconfigurable systems that are not available in the PR tools from the FPGA vendors. The tool is available from [6]. We spent much energy on making it easy to use. Through our effort, we hope to stimulate research on self-aware and self-adaptive systems using FPGAs.

Acknowledgment

This work is supported by the Norwegian Research Council founded project *Context Switching Reconfigurable Hardware for Communication Systems* (COSRECOS) [6], under grant 191156V30.

4. REFERENCES

[1] Xilinx Inc., "Partial Reconfiguration User Guide," 2011, rel 13.2.

[2] Mark Bourgeault, "Alteras Partial Reconfiguration Flow," 2011, available online: `http://www.eecg.utoronto.ca/~jayar/FPGAseminar/FPGA_Bourgeault_June23_2011.pdf`.

[3] A. A. Sohanghpurwala, P. Athanas, T. Frangieh, and A. Wood, "OpenPR: An Open-Source Partial-Reconfiguration Toolkit for Xilinx FPGAs," in *Proc. of the IEEE Int. Symp. on Parallel and Distr. Processing Works. (IPDPSW)*, 2011, pp. 228–235.

[4] D. Koch, C. Beckhoff, and J. Teich, "ReCoBus-Builder– a Novel Tool and Technique to Build Statically and Dynamically Reconfigurable Systems for FPGAs," in *Proc. of Int. Conf. on Field-Progr. Logic and Applications (FPL)*, Sept. 2008, pp. 119–124.

[5] D. Koch, C. Beckhoff, and T. Jim, "GoAhead: A Partial Reconfiguration Framework for Xilinx FPGAs," in *20th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE Computer Society, Apr. 2012.

[6] "Project website: *Context Switching Reconfigurable Hardware for Communication Systems*." [Online]. Available: http://www.mn.uio.no/ifi/english/research/projects/cosrecos/