Overlay Architectures for Heterogeneous FPGA Cluster Management.

Théotime Bollengier, Mohamad Najem, Jean-Christophe Le Lann, Loïc Lagadec

Demo: Overlay Architectures For Heterogeneous FPGA Cluster Management

Théotime BOLLENGIER
b<>com
Brest, France
theotime.bollengier@b-com.com

Mohamad NAJEM, Jean-Christophe LE LANN, Loïc LAGADEC
Lab-STICC MOCS
ENSTA Bretagne
Brest, France
{mohamad.najem, jean-christophe.le_lann, loic.lagadec}@ensta-bretagne.fr

***Abstract*—Overlays are reconfigurable architectures synthesized on commercial of the shelf (COTS) FPGAs. Overlays bring some advantages such as portability, resources abstraction, fast configuration, and can exhibit features independent from the host FPGA. We designed a fine-grained overlay implementing novel features easing the management of such architectures in a cluster of heterogeneous COTS FPGAs. This demonstration shows the use of this overlay in an FPGA cluster, performing a hardware application live migration between two nodes of a cluster. It also illustrates fault tolerance of the cluster.**

I. Introduction

Overlays are reconfigurable architectures implemented on top of FPGAs. They are regular designs described using structural HDL, but have reconfigurable capabilities. They may be considered as "softcore FPGA IPs". Thus, an overlay can be envisioned from two levels of abstraction: *i*) the *functional architecture* is the top view, it is the set of reconfigurable elements available to the applications that target the overlay; *ii*) the *implementation* is the bottom view, it is the way the functional architecture is implemented and synthesized (as a regular IP) on the host FPGA. The functional architecture of an overlay is independent from its implementation and from its host FPGA. Different granularities can be considered: fine-grained overlays are generally composed of LUTs operating at the bit level, while function units of coarser architectures use mathematical operators on words. Overlays are similar to the Java Virtual Machine which enables the exact same bytecode to be executed on different processors (with different instructions sets) for which the JVM has been compiled for.

Thus, overlays have three advantages:

- they can be used to implement the same functional architecture on different hosts, bringing bitstream compatibility between heterogeneous COTS FPGAs;
- the overlay functional architecture may be coarser, simpler, or offer more complex macro blocks than the one offered by the architecture of its host;
- the overlay designers can add features to the overlay implementation that may not be present on the host FPGA, such as dynamic context saving/restoring or configuration pre-loading.

However, compared with a bare metal use of FPGA resources, FPGA virtualization with overlays has a significant cost in terms of resources usage and operating frequency. Therefore, overlays are used in applications for which their advantages justify virtualization cost. Sekanina used coarse grained overlays [1] for evolvable hardware research, benefiting from shorter synthesis and configuration time. Lysecky et al. [2] designed a fine grained overlay with extra routing resources to ease the task of their Just-In-Time synthesizer. To lower compilation time, Coole and Stitt introduced intermediate fabrics [3], which are application specific coarse grained overlays. Dirk Koch et al. [4] integrated a fine-grained overlay in the datapath of a MIPS processor to get a portable custom instructions set softcore processor. Brant and Lemieux addressed the area overhead problem by doing target specific optimizations on the implementation of their fine-grained overlay named ZUMA [5], getting a ratio down to 40 physical LUT per virtual LUT, which is less than one third of a none optimized implementation. Jain et al. [6] also decreased their overlay size down to 70 % and reached frequencies excessing 300 MHz by efficiently using the host DSP blocks in the coarse grained virtual functional units.

Our work aims at virtualizing reconfigurable resources from a heterogeneous set of COTS FPGAs (i.e. from different vendors, and with different characteristics), interconnected as a cluster. Over the lifetime of its infrastructure, components of such an FPGA cluster are gradually updated and replaced (to follow technology evolution over time, and FPGAs sales and trends). This results in FPGAs with different characteristics and from different vendors being used at the same time. However, a bitstream generated for a given FPGA cannot be loaded into a FPGA of a different model. In this context, overlays are relevant as they allow homogenizing these resources: an application design targeting the overlay is no longer tied to a limited set of FPGAs from the cluster, and can run on any node implementing the overlay.

Moreover, overlay implementations can be instrumented to ease the management of such a cluster. In this work, we add novel features to extract or inject the execution state of an overlay, to allow hardware task preemptive scheduling on a node, and application live migration between nodes. This, in turns, allows to perform load balancing or consolidation, to manage application priorities and to provide fault tolerance.

II. Overlay design and use

In this work, we designed a LUT based overlay. Even though fined-grained overlays exhibit an important area and timing overhead, they are more general purpose than coarser architectures which are limited to data-flow applications due to the lack of control structures, and are tied to application domains by the choice of their operators.

The overlay HDL description is automatically generated from a set of architectural parameters. The generated HDL code is portable and can be simulated or implemented on any COTS FPGA. The top part of Fig. 1 shows the synthesis flow from the overlay generation to its physical implementation on FPGA.

Synthesizing an application design to the overlay architecture is done in different steps. First a RTL synthesizer transforms the application description into a netlist composed of latches and arbitrary logic gates. This netlist is then transformed, optimized and mapped to the overlay resources. Next, it is placed and routed on the overlay. Finally, the virtual bitstream is generated by extracting the configuration of each one of the overlay's resources according to the placed and routed netlist. These four synthesis steps are gathered in one step called "synthesis targeting the overlay" at the bottom of Fig. 1.

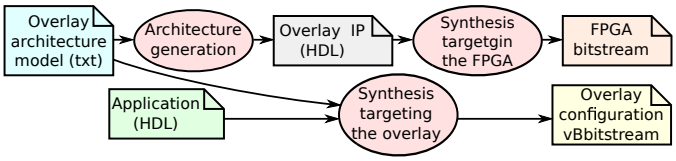

Fig. 1. Two flows: overlay synthesis on the FPGA, and application synthesis on the overlay.

III. Object of the demonstration

We propose a demonstration that illustrates:

- how to offer an homogeneous view of a heterogeneous set of FPGAs;
- the live migration of a hardware application design between two nodes;
- fault tolerance of an overlay cluster.

The setup includes two FPGAs from two vendors (Xilinx and Altera) as compute nodes, and a host PC as a controller. The hardware application design is an image filter. It is synthesized, placed an routed for the proposed overlay. The generated virtual bitstream (vBitstream) is a job to be executed on a compute node. A screen is attached to each FPGA node, displaying the result of the image being filtered, so that the audience can visualize the progress of the job execution. Fig. 2 shows the experimental setup. The two FPGAs are connected to the host PC through Ethernet, each one is attached to an ARM processor running a local hypervisor whose part is to transfer the host management requests to the FPGA. For this demonstration, the processors are also used to display the image being processed.

The first goal is to show the execution of the same application design (from the exact same vBitstream) on two different FPGAs. This is done by deploying the same overlay architecture on both Xilinx and Altera FPGAs. Then the host PC is used to dispatch the vBitstream of the synthesized filter application on both FPGAs.

Live migration is demonstrated by running the filter application on one node, halting the application execution, capturing the execution state of the node's overlay, and then restoring the state on the overlay of the second node. The application resumes on the second node.

Fault tolerance at the cluster level is illustrated by running the filter application on one node. The host controller regularly backups the execution state of the running node. Then the power of the running node is shut down. When the host controller notices that the running node have disappeared (the node does not respong to heartbeat pings anymore), the host sends the vBitstream of the interrupted application along with the last execution state backup to the second node. The execution resumes on the second node at the last backup.

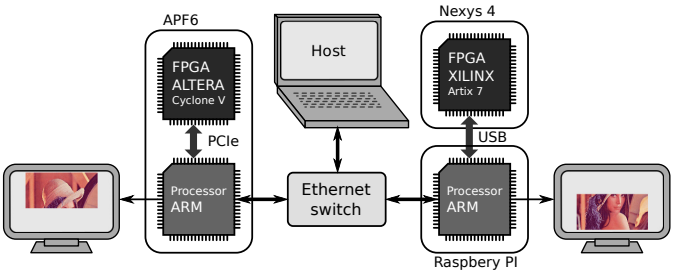

Fig. 2. Experimental setup.

References

[1] L. Sekanina, "Virtual reconfigurable circuits for real-world applications of evolvable hardware," in *Evolvable Systems: From Biology to Hardware, 5th International Conference, ICES 2003, Trondheim, Norway, March 17-20, 2003, Proceedings*, 2003, pp. 186–197.

[2] R. L. Lysecky, K. Miller, F. Vahid, and K. A. Vissers, "Firm-core virtual FPGA for just-in-time FPGA compilation," in *Proceedings of the ACM/SIGDA 13th International Symposium on Field Programmable Gate Arrays, FPGA 2005, Monterey, California, USA, February 20-22, 2005*, 2005, p. 271. [Online]. Available: http://doi.acm.org/10.1145/1046192.1046247

[3] J. Coole and G. Stitt, "Intermediate fabrics: virtual architectures for circuit portability and fast placement and routing," in *Proceedings of the 8th International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2010, part of ESWeek '10 Sixth Embedded Systems Week, Scottsdale, AZ, USA, October 24-28, 2010*, 2010, pp. 13–22. [Online]. Available: http://doi.acm.org/10.1145/1878961.1878966

[4] D. Koch, C. Beckhoff, and G. G. F. Lemieux, "An efficient FPGA overlay for portable custom instruction set extensions," in *23rd International Conference on Field programmable Logic and Applications, FPL 2013, Porto, Portugal, September 2-4, 2013*, 2013, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/FPL.2013.6645517

[5] A. Brant and G. G. F. Lemieux, "ZUMA: an open FPGA overlay architecture," in *2012 IEEE 20th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2012, 29 April - 1 May 2012, Toronto, Ontario, Canada*, 2012, pp. 93–96. [Online]. Available: http://dx.doi.org/10.1109/FCCM.2012.25

[6] A. K. Jain, D. L. Maskell, and S. A. Fahmy, "Throughput oriented fpga overlays using dsp blocks," 2016.