State of the Akvario Project

Andreas Erik Hindborg, Nicklas Bo Jensen, Pascal Schleuniger and Sven Karlsson
DTU Compute
Technical University of Denmark

I. Introduction

The cost of producing a custom ASIC is ever increasing. The prices have long since reached a point where it is not feasible for most research groups to produce a custom ASIC. At the same time, the number of instructions simulated per second by software based simulators for multi-core systems has remained constant between 1998 and 2008 [1]. Further, as the number of simulated cores is increased, the number of instructions executed per simulated core is decreased for a constant overall throughput. In the last decade, researchers have tried to utilize FPGA devices to accelerate simulation. Mid-range FPGA devices have reached a size where 64+ cores can be implemented or simulated in an affordable device (sub $1000) [2]. We have developed Akvario, a toolset that helps researchers evaluate future many-core systems by providing easy access to the power of FPGA devices. Akvario is based on the Tinuso processor core and other IP cores we have developed.

Tinuso is a processor architecture that targets implementation in FPGA fabric. Tinuso-I is the first implementation of the Tinuso architecture. It is a high performance processor core, achieving clock rates of up to 376 MHz when synthesized for modern FPGA devices [3]. Tinuso-I is smaller and faster than other state-of-the-art processor cores that target implementation in FPGA fabric [4]. Akvario includes a set of tools that allow researchers and designers to create highly customized Tinuso-based many-core systems for FPGA devices. We use Akvario to construct accelerators for demanding high-performance compute systems [5], [6]. This is possible because the Tinuso architecture is designed with performance in mind, and because the Tinuso architecture is highly configurable. We also use Akvario to construct simulation platforms that mimic future many-core systems as known from Intel SCC [7] and Adapteva Epiphany.

When we use Akvario as a basis for a simulator, it differs from other FPGA based simulators such as HASim and RAMP Gold by having a 1:1 mapping of host cycles to simulated target cycles. This direct RTL implementation strategy makes it difficult to change the Tinuso core to explore new micro-architectural features. However, it is ideal for exploring higher-level concepts such as programming models and languages for future many-core systems. It also allows Tinuso based systems achieve higher performance than other FPGA based simulators. HASim is capable of simulating a 5-stage in-order pipeline at 5.1 MIPS [8], while a Tinuso-based system can achieve more than 300 MIPS.

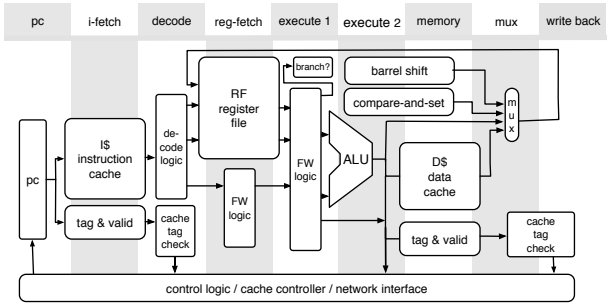

Fig. 1. The Tinuso processor core pipeline.

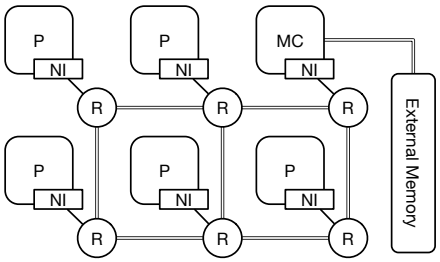

Fig. 2. Tinuso cores in a mesh network.

II. State of the Akvario IP Cores and Tools

Tinuso-I is a statically scheduled, in-order processor core that targets high performance when synthesized for FPGA fabric. The processor core relies on pipelined access to caches and register files to achieve a high clock rate. See figure 1 for reference. The deep pipeline results in 4 branch delay slots. Tinuso-I supports predication on most instructions to be able to place conditional code in these branch delay slots.

The instruction set for the processor core can be highly customized. Features such as multiplier, different bit-shifters and IEEE-754 compliant floating point unit can be enabled or disabled when configuring the core. Further, Tinuso-I supports a tightly integrated co-processor interface. The user can add additional instructions or hardware blocks via this interface.

The Tinuso-I core is designed to connect to a network interface for an on-chip network. See figure 2 for reference. We currently support a 2D mesh network with xy routing, but other topologies are possible.

Akvario systems are easily constructed by describing the desired system topology with an XML-based configuration file. A system construction tool parses the XML file and

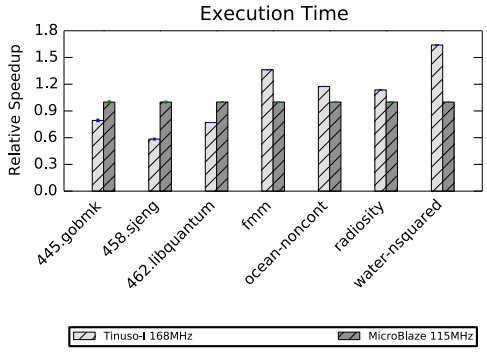

Fig. 3. Execution times for benchmarks executed on a Tinuso processor core and a MicroBlaze processor core.

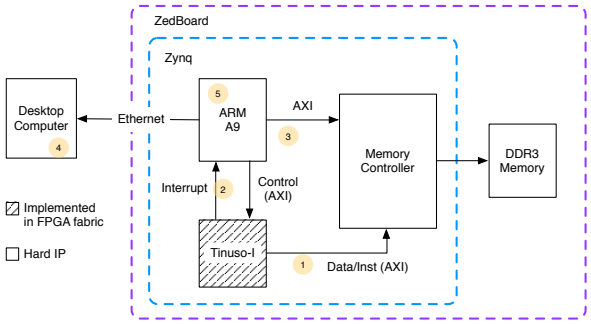

Fig. 4. Tinuso provides access to operating system services via offloading.

constructs RTL code for the system. The tools are even capable of doing a full implementation flow for specific Xilinx FPGA devices by invoking the Xilinx Vivado tool automatically [9].

We use a custom port of the GCC compiler to compile programs for our Tinuso systems. The compiler is very efficient in utilizing the predicated instructions in the Tinuso instruction set to increase performance in code that contain a high amount of branches [10].

We currently support programs written in the C language. We use a port of the Newlib C library to provide a standard C runtime for programs running on Tinuso systems. We experimentally support the C++ language.

At the moment the Akvario tools support a simple message-passing programming model (a subset of MPI), and a shared memory programming model based on software assisted release consistency.

Akvario systems are usually considered resource constrained embedded systems. In order to provide access to complex services such as file systems and process management, we have added a system call offloading capability to the Akvario tools. See figure 4 for reference. This technique allow programs running on Tinuso cores to access operating system services on a local desktop machine via an RPC mechanism [11]. The current implementation of this mechanism relies on the ARM Cortex A9 processor core present in the Xilinx Zynq devices. With this offloading mechanism we have run benchmarks from the SPEC 2006 and SPLASH2 benchmark suites, see figure 3. We are able to show a speedup of up to 64% for Tinuso-I versus Xilinx MicroBlaze.

III. Current and Future Efforts

We are constantly improving IP cores and tools. At the moment we are working to release the Tinuso-I RTL model and associated tools under a **permissive open source license**. We plan to expand the set of supported programming languages to include specialized languages such as OpenMP, OpenCV and RVC-CAL.

IV. Conclusion

Akvario is a versatile toolbox for implementing many-core accelerators in FPGA fabric. The tools allow researchers to study future many-core systems and to construct accelerators for demanding compute problems. We expect Akvario to be released to the public domain in the near future.

Acknowledgments

This work has been partially funded by the ECSEL JU as part of the COPCAMS project under GA number 332913.

References

[1] Z. Tan, A. Waterman, H. Cook, S. Bird, K. Asanović, and D. Patterson, "A Case for FAME: FPGA Architecture Model Execution," in *International Symposium on Computer Architecture (ISCA)*. ACM, 2010.

[2] Z. Tan, A. Waterman, R. Avizienis, Y. Lee, H. Cook, D. Patterson, and K. Asanovic, "RAMP gold: An FPGA-based architecture simulator for multiprocessors," in *Design Automation Conference (DAC)*, June 2010.

[3] P. Schleuniger, S. McKee, and S. Karlsson, "Design Principles for Synthesizable Processor Cores," in *Architecture of Computing Systems*, ser. Lecture Notes in Computer Science. Springer, 2012, vol. 7179.

[4] P. Schleuniger, "Performance Aspects of Syntheizable Computing Systems," Ph.D. dissertation, Technical University of Denmark, 2014.

[5] P. Schleuniger, A. Kusk, J. Dall, and S. Karlsson, "Synthetic aperture radar data processing on an FPGA multi-core system," in *Architecture of Computing Systems*. Springer, 2013.

[6] P. Schleuniger and S. Karlsson, "A Synthesizable Multicore Platform for Microwave Imaging," in *10th International Symposium on Applied Reconfigurable Computing*. Springer, 2014.

[7] T. Mattson, R. van der Wijngaart, M. Riepen, T. Lehnig, P. Brett, W. Haas, P. Kennedy, J. Howard, S. Vangal, N. Borkar, G. Ruhl, and S. Dighe, "The 48-core SCC Processor: the Programmer's View," in *Supercomputing (SC)*, Nov 2010.

[8] M. Pellauer, M. Vijayaraghavan, M. Adler, Arvind, and J. Emer, "Quick Performance Models Quickly: Closely-Coupled Partitioned Simulation on FPGAs," in *International Symposium on Performance Analysis of Systems and software (ISPASS)*, April 2008.

[9] A. Hindborg, P. Schleuniger, N. B. Jensen, M. Walter, L. Brock-Nannestad, L. Bonnichsen, C. W. Probst, and S. Karlsson, "Automatic Generation of Application Specific FPGA Multicore Accelerators," in *The Asilomar Conference on Signals, Systems, and Computers*, 2014.

[10] N. B. Jensen, P. Schleuniger, A. Hindborg, M. Walter, and S. Karlsson, "Experiences with Compiler Support for Processors with Exposed Pipelines," in *IEEE International Parallel & Distributed Processing Symposium: Reconfigurable Architectures Workshop (RAW)*, 2015, to appear.

[11] A. E. Hindborg, P. Schleuniger, N. B. Jensen, and S. Karlsson, "Hardware Realization of an FPGA Processor – Operating System Call Offload and Experiences," in *8th Conference on Design & Architectures for Signal & Image Processing (DASIP)*, 2014.