Introduction to Special Issue on Reconfigurable Components with Source Code

ANDRÉ DEHON, University of Pennsylvania
DEREK CHIOU, University of Texas

CCS Concepts: • **Computer systems organization** → **Reconfigurable computing**; • **Hardware** → **Reconfigurable logic and FPGAs**;

Additional Key Words and Phrases: Design reuse, source code

ACM Reference Format:

André DeHon and Derek Chiou. 2016. Introduction to special issue on reconfigurable components with source code. *ACM Trans. Reconfigurable Technol. Syst.* 9, 3, Article 19 (May 2016), 2 pages.

DOI: <http://dx.doi.org/10.1145/2907949>

As Field-Programmable Gate Array (FPGA) capacity increases to hold larger designs, our FPGA systems and designs become more ambitious. It is increasingly useful to share and reuse larger blocks and components. FPGA conferences and journals report on designs, but few of these are widely available for community-wide reuse. At the same time, with the decreasing cost of digital data storage and digital libraries becoming the primary storage and distribution for our journal, it becomes possible to capture, store, and distribute design artifacts along with the journal articles. To address the need for shareable, reusable components and as a first experiment with reviewing and including design artifacts, TRETs organized this special issue that sought articles accompanied by widely usable source code for key reconfigurable building blocks.

Evaluating the design artifacts and their usability is a key challenge. The tool chains for the most widely used, mainstream FPGAs are proprietary, requiring licensing. The technology advances rapidly, and vendors regularly supply new versions of tools that behave differently, typically better, than earlier versions. It can be a major effort to assemble and install the tools necessary to process a Register-Transfer Language (RTL)-level FPGA design. To make the process manageable for the reviewers and for the potential users of these designs, we leveraged the FPGA Accelerator Research Infrastructure Cloud (NSF Grant No. 1205721, see openfabric.org). FABRIC provides a common development platform, including CAD tools and servers on which to run those tools, as well as FPGA platforms, for (i) the developers of the designs, (ii) the reviewers of the designs, and (iii) the users of the designs.

Emphasis in the review and editorial revision of these articles differed in a few ways from the standard articles to accommodate the design artifact and the different goals. We demanded that the design come with a clearly stated, open-source license. Articles should explain clearly how to integrate and use the design. We looked for papers that described useful, usable, and accessible designs. We asked the reviewers to assure

Authors' addresses: A. DeHon, Department of Electrical and Systems Engineering, 200 S. 33rd St., Philadelphia, PA, 10104; email: andre@acm.org; D. Chiou, The University of Texas at Austin, 2501 Speedway, ENS Building, room 540, C0803, Austin, TX 78712; email: derek@ece.utexas.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

2016 Copyright is held by the owner/author(s).

1936-7406/2016/05-ART19

DOI: <http://dx.doi.org/10.1145/2907949>

that the designs would build with the openfabric.org tools and run supplied tests. These designs are often not the first instances of an FPGA design solving a particular problem, but they are mature, well-documented designs with rich parameterization options. Each of these designs comes with documentation on the tools they require and make build targets to automate the tool flow and verification operations.

The four designs in this issue are building blocks that underlie a wide set of algorithms and reconfigurable computations.

A key benefit of implementing custom datapaths on FPGAs with fine-grained configurable logic is the ability to tailor the computation to the exact needs of the problem. VFLOAT provides an open-source, highly parameterizable floating-point library with primitive floating-point operators that can be configured to specified precisions and ranges. Fang and Leiser describe the latest extensions to the VFLOAT library with an emphasis on the design and characterization of the reciprocal, divide, and square-root units.

While modern FPGAs provide embedded Random Access Memories (RAMs) with two memory ports, many designs demand a large and varying number of ports. A number of techniques for efficiently building multiport memories on top of the fixed-port memories in FPGAs have emerged over the past few years. Abdelhadi and Lemieux describe designs for a wide-range of multi-ported memories bringing together a suite of techniques to provide efficient implementations optimized for various scenarios and providing guidance on which organizations are best suited for particular multi-ported memory requirements.

High-speed FPGA arithmetic is highly pipelined. This can be used for high-speed data accumulation as is needed in many signal processing and scientific computing tasks. However, managing feedback paths to keep highly pipelined adders operating at full throughput can be challenging. Wilson and Stitt bring together several techniques to provide parameterized accumulator designs that cover a wide range of design tradeoffs.

Efficient designs for FPGAs carefully coordinate data movement and reuse. Window filters are a common computational pattern that can be highly optimized for streaming computations on FPGAs. Stitt, Schwartz, and Cook provide an efficient framework for assembling window filters that allows configurable parallelism as well as window and image size.

Capturing design artifacts, promoting community and design reuse, and promoting repeatability of experiments are all of great interest to our community and the ACM in general. We hope this is the beginning of a trend toward ubiquitous design sharing. Best practices for capturing, reporting, and archiving designs are still evolving. Please provide feedback and join the ongoing discussion. If you build on this work, then cite the TREATS articles and send the authors your feedback. Let the TREATS editors know what you would like to see in this area. The ACM/SIGDA TCFPGA is one focal point for this discussion you can join; see their wiki wiki.tcfpga.org. Finally, show your support by distributing your own designs and submitting your design articles to TREATS.

Received March 2016; accepted March 2016