Guest Editors' Introduction

Agile and Open-Source Hardware

Yungang Bao
Chinese Academy of Sciences

Trevor E. Carlson
National University of Singapore

As the benefits of traditional technology scaling, like Dennard Scaling and Moore's Law, slow significantly, computer architecture is poised to enter a golden age of innovation. Domain-specific architectures (DSA) are a promising solution to continue improving computing performance, while maintaining the level of energy- and area-efficiency previously found in technology scaling. Unfortunately, traditional methodologies of chip design and hardware development have created significant barriers, requiring extremely high non-recurring engineering costs in tools, labor, IPs, and time, which ultimately hold back broad adoption of DSA.

In contrast, the significant engineering costs and extremely long design cycles for software development have reduced significantly over the last few decades due to the proliferation of open-source software and the use of agile software development techniques. Small teams of software developers can now realize their innovative ideas quickly using higher level abstractions and tools that are developed by their own user base. Inspired by these impressive results from the software community, agile and open hardware design is considered to be one of the most promising ways to lower the design cost of chip design, although there are still many challenges in abstraction, methodologies, and tools. This Special Issue on Agile and Open-Source Hardware is to record recent progress in this emerging field and also encourage the community to engage proactively in this area.

Small teams of software developers can now realize their innovative ideas quickly using higher level abstractions and tools that are developed by their own user base. Inspired by these impressive results from the software community, agile and open hardware design is considered to be one of the most promising ways to lower the design cost of chip design, although there are still many challenges in abstraction, methodologies, and tools.

This special issue consists of 13 articles, which cover a variety of research topics related to fast, agile, and open hardware design, including methodologies, languages, abstractions, and simulators. Below, we briefly outline the articles included in this special issue, which include topics in ASIC and FPGA design methodologies, hardware description languages and simulation, and specifications and hardware generators.

ASIC DEVELOPMENT FRAMEWORKS

The article by Amid *et al.*, titled "Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs," presents the Chipyard framework, an integrated SoC design, simulation, and implementation environment for specialized compute systems, in order to address

Digital Object Identifier 10.1109/MM.2020.3002606
Date of current version 30 June 2020.

Published by the IEEE Computer Society

the challenges in integration and validation of complex SoCs. The article describes two main highlights of the Chipyard framework: one is the use of configurable, composable, open-source, generator-based IP blocks that can be used across multiple stages of the hardware development flow while maintaining design intent and integration consistency; another is cloud-hosted FPGA-accelerated simulation and rapid ASIC implementation, which enable continuous validation of physically realizable customized systems.

The article by Balkind *et al.*, titled "OpenPiton at 5: A Nexus for Open and Agile Hardware Design," shares some of the lessons learned during the development of OpenPiton over the past five years. OpenPiton was first designed to perform cutting-edge computer architecture research at Princeton University and has led to thousands of downloads and numerous academic publications after its opening to the public. The article provides examples of how OpenPiton has been used to efficiently test novel research ideas and discusses how OpenPiton has evolved due to its open development and feedback from the open-source community.

In the article "CHIPKIT: An Agile, Reusable Open-Source Framework for Rapid Test Chip Development," Whatmough *et al.* describe the CHIPKIT framework, a reusable SoC subsystem that provides basic IO, an on-chip programmable host, off-chip hosting, memory, and peripherals. This system can be readily extended with new IP blocks to generate custom test chips. Central to CHIPKIT is an agile RTL development flow, including a code generation tool called VGEN. The article also describes a front-to-back design example, drawing on multiple generations of successful test chips designed at Harvard.

FPGA DEVELOPMENT FRAMEWORKS

The article by Tang *et al.*, titled "OpenFPGA: An Open-Source Framework for Agile Prototyping Customizable FPGAs," introduces OpenFPGA, an open-source framework that can automate and significantly accelerate the development cycle of customizable FPGA architectures. OpenFPGA framework consists of two design flows: the production flow, which can translate an XML-based FPGA architecture description to a complete GDSII layout; the end-user flow allows FPGA developers to convert Verilog applications to configuration bitstreams for use on the custom FPGA fabric. This article demonstrates that OpenFPGA achieves less than 24-h layout generation of two FPGA fabrics using a commercial 12-nm standard-cell library and 40-nm custom cells, respectively.

In the article "SymbiFlow and VPR: An Open-Source Design Flow for Commercial and Novel FPGAs," Murray *et al.* discuss the SymbiFlow project, which seeks to create an open-source CAD flow for FPGAs that can be used not only to program commercial FPGAs but also to evaluate new FPGA architectures. The article proposes a data-driven approach that uses highly adaptable and re-targetable opensource tools which aims to achieve specificity and flexibility. Using this approach, SymbiFlow can fully map designs to the commercial Xilinx Artix 7 devices and to other (existing or novel) FPGAs as well.

HARDWARE DESCRIPTION LANGUAGES AND SIMULATION

The article by Jiang *et al.*, titled "PyMTL3: A Python Framework for Open-Source Hardware Modeling, Generation, Simulation, and Verification," presents PyMTL3, a Python framework for open-source hardware modeling, generation, simulation, and verification. In addition to compelling benefits from using the Python language, PyMTL3 is designed to provide flexible, modular, and extensible workflows for both hardware designers and computer architects. The article argues that PyMTL3 can help to jump-start the open-source hardware ecosystem with three main features: it is embedded in Python; it emphasizes interoperability with other open-source hardware tools; promotes agile and test-driven design methodologies.

In the article "LiveHD: A Productive Live Hardware Development Flow," Wang *et al.* propose LiveHD, an open-source framework for incremental hardware synthesis and simulation that provides feedback within seconds. LiveHD follows three principles for design automation.

LiveHD follows three principles for design automation, i.e., dividing the job into partition regions or checkpoints, incrementally transforming these partition regions where code change happens and reloading the partition regions into a running program without restarting.

The article demonstrates that LiveHD can provide feedback for synthesis, placement, and routing in less than 30 s for most changes tested with negligible QoR impact on the quality of results. For the incremental simulation, LiveHD is capable of getting any simulation cycle in under 2 s for a 256 RISC-V core design.

The article by Dangwal *et al.*, titled "Agile Hardware Development and Instrumentation With PyRTL," presents PyRTL, an open-source Python-based hardware development toolkit, which provides a pathway to concisely and precisely design hardware structures. The article describes how PyRTL uses a small and well-defined internal core structure that provides a minimal set of hardware primitives to enable hardware reuse using modern object-oriented programming features. The article presents an examination of its custom intermediate representation for hardware debugging, analysis, and instrumentation. PyRTL has been shown to be useful in supporting fast design iteration in a variety of domains including cryptography and machine learning.

In the article "LastLayer: Toward Hardware and Software Continuous Integration," Vega *et al.* present LastLayer, an open-source tool that enables hardware and software continuous integration and simulation. LastLayer can integrate Verilog designs into any programming language that supports the C foreign function interface and allows external programs convenient access to storage resources such as registers and memories in the design as well as control over the hardware simulation. Moreover, LastLayer achieves this software integration without requiring any hardware modification. This article presents evaluations of two representative integration examples: a hardware adder written in Verilog operating over NumPy arrays, and a ReLu vector-accelerator written in Chisel processing tensors from PyTorch.

The article by Scott Beamer, titled "A Case for Accelerating Software RTL Simulation," studies the performance bottleneck of RTL simulators by profiling with hardware performance counters. The article compares open-source simulators with a leading commercial simulator and demonstrates that open-source simulators not only outperform the commercial simulator but also achieve comparable or higher instruction throughput on their platform. The article further reveals there is room to improve simulation performance and presents a road map of potential research directions.

From processors to accelerators, simulation to design, and languages to methodologies, this special issue covers an important slice of the current state of open and agile tools.

SPECIFICATIONS AND GENERATORS

In the article "Generating Systolic Array Accelerators With Reusable Blocks," Jia *et al.* investigate how to automatically generate systolic arrays that are widely used in spatial hardware designs and well-suited for many tensor-based applications. The article analyzes the systolic array design space and identifies the common structures of different systolic dataflows. The article further presents how to build hardware module templates reused for different dataflows and a systolic array generator that transforms tensor algorithm definitions into a complete systolic hardware architecture. Experiments show that the proposed approach can implement systolic array designs for different applications and dataflows with little engineering effort, and the performance throughput outperforms HLS designs.

The article by Petrisko *et al.*, titled "BlackParrot: An Agile Open-Source RISC-V Multicore for Accelerator SoCs," introduces BlackParrot, which aims to be the default open-source, Linux-capable, cache-coherent, 64-bit RISC-V multicore. The goal of BlackParrot is to be community-driven, infrastructure agnostic, and a multicore which is Pareto optimal in terms of power, performance, area, and complexity. The article describes how the development of BlackParrot is guided by three core principles: be tiny, be modular, and be friendly. BlackParrot has been validated with a GlobalFoundries 12-nm FinFET tapeout and can be used as a standalone Linux processor or as a malleable fabric for an agile accelerator SoC design flow.

In the article "Tydi: An Open Specification for Complex Data Structures Over Hardware Streams," Peltenburg *et al.* introduce Tydi, an open specification that allows developers to map

composite and dynamically sized data structures onto hardware streams via a stream-oriented specification and type system. Tydi furthermore provides an abstract, hardware-oriented view of data structures. This helps designers to lift the abstraction of streaming dataflow designs, reducing design effort. The type system aims to allow complex data structures to be as easy to use in streaming dataflow designs as in modern software languages today.

From processors to accelerators, simulation to design, and languages to methodologies, this special issue covers an important slice of the current state of open and agile tools. Learning from software design's fast turn-around cycles and levels of abstraction, we hope that these works demonstrate how this nascent field is quickly maturing, enhancing the capabilities of today's hardware designers and researchers to produce high-quality, game-changing designs in much less time than before.

Yungang Bao is a Professor at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). His research interests include open-source hardware and agile chip design, datacenter architecture and memory systems. His research work such as Labeled von Neumann Architecture (LvNA), Hybrid Memory Trace Tool (HMTT), and PARSEC 3.0 has been adopted by the industry (e.g., Alibaba, Huawei and Intel etc.) and the research community. Contact him at baoyg@ict.ac.cn.

Trevor E. Carlson is an Assistant Professor at the National University of Singapore (NUS). His research interests include fast, scalable, and open simulation methodologies; performance modeling; efficient microarchitecture; and accelerator design. He co-developed the Sniper Multi-Core Simulator, which is being used by hundreds of researchers in academia and industry, to evaluate the performance and power-efficiency of next generation systems. Contact him at tcarlson@comp.nus.edu.sg.