A Comparative Survey of Open-Source Application-Class RISC-V Processor Implementations

Alexander Dörflinger
Mark Albers
Benedikt Kleinbeck
Yejun Guan
Harald Michalik
doerflinger,albers,kleinbeck,guan,michalik
@ida.ing.tu-bs.de
Institute of Computer and Network Engineering (IDA)
Technische Universität Braunschweig
Braunschweig, Germany

Raphael Klink
Christopher Blochwitz
Anouar Nechi
Mladen Berekovic
klink,blochwitz,nechi,berekovic@iti.uni-luebeck.de
Institute of Computer Engineering (ITI)
Universität zu Lübeck
Lübeck, Germany

ABSTRACT

The numerous emerging implementations of RISC-V processors and frameworks underline the success of this Instruction Set Architecture (ISA) specification. The free and open source character of many implementations facilitates their adoption in academic and commercial projects. As yet it is not easy to say which implementation fits best for a system with given requirements such as processing performance or power consumption. With varying backgrounds and histories, the developed RISC-V processors are very different from each other. Comparisons are difficult, because results are reported for arbitrary technologies and configuration settings. Scaling factors are used to draw comparisons, but this gives only rough estimates. In order to give more substantiated results, this paper compares the most prominent open-source application-class RISC-V projects by running identical benchmarks on identical platforms with defined configuration settings. The Rocket, BOOM, CVA6, and SHAKTI C-Class implementations are evaluated for processing performance, area and resource utilization, power consumption as well as efficiency. Results are presented for the Xilinx Virtex UltraScale+ family and GlobalFoundries 22FDX ASIC technology.

CCS CONCEPTS

• **Computer systems organization** → System on a chip; **Serial architectures**.

KEYWORDS

RISC-V, application-class, open-source, FPGA, ASIC, GlobalFoundries 22FDX, Virtex UltraScale+, benchmarks, energy efficiency

Permission to make digital or hard copies of all or part 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 components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CF '21, May 11–13, 2021, Virtual Conference, Italy
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8404-9/21/05…$15.00
https://doi.org/10.1145/3457388.3458657

ACM Reference Format:
Alexander Dörflinger, Mark Albers, Benedikt Kleinbeck, Yejun Guan, Harald Michalik, Raphael Klink, Christopher Blochwitz, Anouar Nechi, and Mladen Berekovic. 2021. A Comparative Survey of Open-Source Application-Class RISC-V Processor Implementations. In *Computing Frontiers Conference (CF '21), May 11–13, 2021, Virtual Conference, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3457388.3458657

1 INTRODUCTION

One decade after the RISC-V project initiation by UC Berkeley, its application area is not limited to academia anymore and the ISA specification [41] is also being widely adopted by industry [38]. In the past few years, a large number of both proprietary and open-source RISC-V implementations emerged. Furthermore, RISC-V ecosystems have been developed to provide software compilers, System-on-Chip (SoC) peripherals and other components, simplifying the generation of FPGA- or ASIC-based RISC-V processor systems. The free and open character of many RISC-V implementations allows reuse of the collaborative open-source projects. Project-specific requirements can be satisfied through custom modifications and extensions. This makes RISC-V particularly interesting for special purpose and niche applications. For instance, RISC-V is a promising architecture for the space domain with stringent reliability requirements [24, 25].

Each ISA implementation has its strengths and weaknesses, making it difficult to select the best-fitting RISC-V solution for a project with dedicated requirements such as performance, power consumption, or simplicity. Research groups typically report results of their implementations for a specific ASIC technology tapeout or FPGA implementation. As the selected technology heavily affects processor speed and power consumption, only a rough indirect comparison is feasible by assuming scaling factors. Different benchmarks are utilized for performance estimations, which again complicates a direct comparison. Furthermore, architectural design parameters (e.g., cache sizes) are defined by each group and publication differently, affecting reported area, power, and performance results.

The main contributions of this work are an analysis and comparison of the most popular application-class open-source RISC-V implementations by running same benchmarks on identical hardware platform. Hereby, an FPGA of the Xilinx Virtex UltraScale+ family

is selected as an evaluation platform, featuring a state-of-the-art FPGA technology. Performance, area, and power measurements are taken for SoC designs and standalone RISC-V cores separately. Additionally, all cores are synthesized for the GlobalFoundries 22FDX ASIC technology. The comparison is based on equal architectural design parameters. Strengths and weaknesses of respective processor cores are discussed, which helps selecting an available RISC-V implementation for academic and commercial projects with specific requirements.

This work concentrates on application-class RISC-V processors covering the medium to high performance range and excludes lightweight RISC-V implementations. Application-class processors typically provide support for UNIX-based Operating Systems (OSs), which brings several advantages. Firstly, it simplifies software development, because one can utilize existing libraries, drivers, and programs. Secondly, memory management and isolation of user programs allows concurrent execution of multiple threads. On the other hand, the OS considerably increases the hardware complexity of the processor [35]. The hardware has to provide three privilege levels (M/S/U-Mode) [40]. Furthermore, the OS demands the A-extension containing atomic load-reserved/store-conditional (LR/SC) instructions and Atomic Memory Operations (AMOs) [41]. A virtual address space requires hardware support for fast address translation, which adds a Transaction Lookaside Buffer (TLB) and Page Table Walker (PTW) to the system. As the OS itself already requires several megabytes of memory, application-class processors typically connect to off-chip memory. The efficiency of memory accesses then relies on the implemented memory hierarchy with caching mechanisms.

The rest of this paper is organized as follows. Sect. 2 provides an overview of previous RISC-V classification and comparison approaches and Sect. 3 presents existing open-source application-class RISC-V implementations. The FPGA and ASIC evaluation platforms are described in Sect. 4. Results of performance, area, power consumption, and energy efficiency are presented in Sect. 5.

2 RELATED WORK

The RISC-V community maintains and steadily updates a list of available cores and SoCs [10]. The provided information is an appropriate starting point for further analysis and comparison of existing implementations. However, it does not guarantee completeness and in particular some smaller RISC-V projects are not listed. Furthermore, it collects only a handful of characteristics and lacks important criteria such as area and performance estimations.

Other works compare RISC-V cores in more detail. [36] describes a tool for exploring RISC-V projects. A tutorial teaches how to use their IDE for running tests and benchmarks on RISC-V soft-cores. However, only the non Linux-capable PicoRV32 [8] core has been integrated and no comparisons to other cores are presented. [23] compares the ultra-low-power cores Zero-riscy, Micro-riscy, and Riscy. It analyzes the core area for the UMC 65 nm technology and calculates power and energy consumption for different workloads. The comparison focuses only on lightweight RISC-V cores targeting low-power applications.

An extensive comparison of 32 bit RISC-V cores is performed in [28] by utilizing the TaPaSCo framework [30]. Maximum operating frequency, resource utilization, and various benchmark scores are measured for eight open-source cores across four FPGA platforms. However, the TaPaSCo framework exhibits some restrictions on the comparison such as technology (FPGA only), ISA (32 bit only), and omission of the L1 cache architectures.

There exist several further survey works comparing multiple RISC-V implementations [29], [37], [34] or processors of different ISAs [17], [32]. However, all are limited to 32 bit variants and target FPGA applications with soft-core processors only. A comparison of cores of the medium to high end performance range is still missing. This work tries to fill this gap and additionally evaluates the readiness of RISC-V cores for ASIC implementations.

3 ANALYSIS OF RISC-V IMPLEMENTATIONS

The RISC-V project overview [10] currently lists 89 cores and further SoC platforms and SoCs. These numbers already present a large variety of ISA implementations, yet it is not fully complete and constantly growing. This work evaluates midrange to high performance cores that satisfy the terms *application-class* and *open-source*, which narrows the selection down. In this context, a RISC-V implementation satisfies the criterion *application-class* if it complies to the RV64I ISA base [41] with a word size of 64 bit and if it is capable to boot a UNIX-based OS. The implementation is *open-source*, if it is published under a license that allows commercial use without imposed fees. There exist open source licenses with significant differences (e.g., copyleft vs. permissive). If not noted otherwise, all of the RISC-V projects analyzed in this work are published under permissive licenses with similar terms and conditions.

The above definition excludes Linux-capable RV32I cores such as the portable RVSoC [35], the FPGA friendly VexRiscv [14], and the Out-of-Order (OoO) RSD [33]. Furthermore, the comparison in this work omits proprietary implementations, namely the RV64GC multi-core within the PolarFire SoC (Microsemi, [9]), customized A25 and AX25 SoCs (Andes Technology, [1]), the SCR5 and SCR7 (Syntacore, [11]), several core complexes from CloudBear [3], and the Bk7 from Codasip [4]. T-head of the Alibaba Group claims to outperform any other RISC-V implementation with its XuanTie-910 processor [20]; however, it is also not available as open source.

Fig. 1 illustrates the academic impact, community activity, and technology support of major open-source application-class RISC-V implementations. It compares the number of Google Scholar hits[1] and the repository activity (number of contributors). Furthermore, the number of supported FPGA evaluation boards and tapeouts has been counted. While tapeouts are typically well documented (see Sect. 3.1 to 3.4), it is more intricate to assess FPGA board support. It is provided through particular project branches or different frameworks (e.g., lowRISC, Si-Five Freedom, OpenPiton). Due to the separation from the main development branch, many FPGA projects rely on out-dated processor versions.

The area of the covered polygon is an indicator for the degree of attention of the respective implementation. Rocket [16] dominates 3 out of 4 categories, which emphasizes its success in both commercial and academic projects. It is followed by CVA6 (formerly named Ariane) [44], whose design has been verified on various FPGA

[1] In order to limit the search to RISC-V relevant results, the term "risc-v" "<name of core>" has been used. Accessed: 2020-12-22.

boards and through several tapeouts. The high number of Google Scholar hits for BOOM [18] denotes its academic importance as an OoO processor. The SHAKTI C-Class processor [27] is maintained by a slightly smaller community than BOOM and CVA6 and counts two tapeouts. The mor1kx [7] of the OpenRISC project published under a weak copy-left license scores with an outstanding support for FPGA boards (provided through the "LED to believe" project). However, its academic impact is small compared to others and there has been no recent contribution activity. Both RiscyOO (also named riscy-OOO) [45] and AnyCore [22] succeeding FabScalar [21] are smaller projects with only 2 or 5 code contributors, have not been taped out yet[2], and are not actively maintained. Fig. 1 is not an exhaustive list of open-source application-class RISC-V projects. However, all others (e.g. Lizard [6]) are excelled by the leading projects presented here.

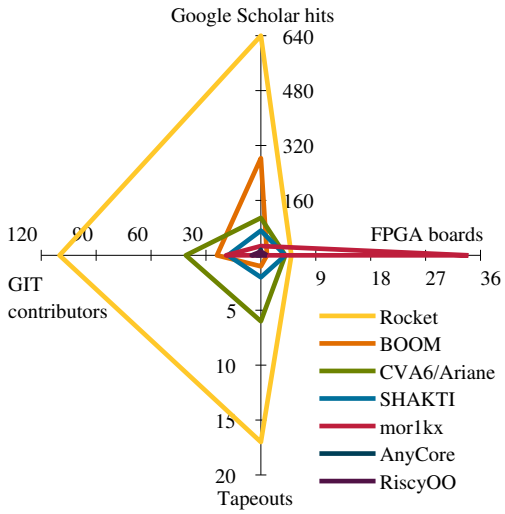

Figure 1: Academic impact, community activity, and technology support of open-source application-class RISC-V processor cores.

We selected the four most prominent implementations for further single-core evaluation and comparison, whereas all of them also offer multicore configurations. The following subsections present the main characteristics of each processor core and its implementation framework. Performance, area, and power efficiency results of previous works are collected.

3.1 Rocket

Rocket is an in-order scalar processor developed at UC Berkeley that provides a 5-stage pipeline: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Writeback (WB). It offers both the RV64G or RV32G variants of the RISC-V ISA and is written in the Chisel Hardware Description Language (HDL) based on object-oriented Scala. The high abstraction level of Chisel allows an easy and mainfold processor customization such as optional activation of ISA extensions (M, A, F, D).

[2]The AnyCore project reports a tapeout for PISA ISA, but not RISC-V ISA.

The branch prediction within the frontend is configurable and provided by a Branch Target Buffer (BTB), Branch History Table (BHT), and Return Address Stack (RAS). The load-store architecture can be configured with a blocking or non-blocking L1D cache. A Memory Management Unit (MMU) supports page-based virtual memory. The execution pipeline holds five functional units, amongst them an integer Arithmetic Logic Unit (ALU) and an optional IEEE 754-2008-compliant Floating-Point Unit (FPU). A Rocket Chip Coprocessor (RoCC) interface is provided for attachment of customized accelerators or coprocessors.

To compose Rocket cores, caches, and interconnects into an integrated SoC, the open-source SoC design generator Rocket Chip Generator [16] can be used. It is integrated within the open-source Chipyard framework, which contains a large set of tools for developing, simulating, and compiling both hardware and software. The framework provides several example configurations (e.g., tiny ... large, single-/multicore) with predefined cache- and predictor settings. Arbitrary peripherals and accelerators may be added to a configuration. A sophisticated simulation platform called FireSim running on Amazon EC2 F1 instances facilitates new developments and adaptions of Chipyard's SoC designs.

There have been numerous tapeouts starting in 2012 with EOS14 (IBM 45 nm SOI, dual-core, 1.5 GHz, 0.9 V), over Raven-3 (ST 28 nm FD-SOI, single-core, 1.3 GHz), up to SiFive U54 [39] (TSMC 28 nm HPC, quad-core, 1.5 GHz). The latter one is offered by SiFive as one of several pre-configured customizable IP cores.

3.2 BOOM

BOOM is a superscalar OoO processor implementing the RV64GC variant of the RISC-V ISA that can be instantiated as a replacement of the Rocket core. Analogously to Rocket, the BOOM core is written in the Chisel HDL and integrated into the Chipyard framework.

The current release named "SonicBOOM" is the fastest publicly available open-source RISC-V core by Instructions per Cycle (IPC) count [46]. Hereby recent works on the BOOM design illustrate the great progress that is still ongoing for RISC-V. Compared to BOOMv2 [19], the BOOMv3 design (SonicBOOM) [46] utilized in this work more than doubles the benchmark scores.

BOOM implements a complex 10-stage pipeline structure with a 12 cycle branch-mispredict penalty. The frontend features a customizable banked L1I cache, TLB, and a decode stage. It contains a sophisticated but also highly configurable branch prediction unit with a fast Next-Line Predictor (NLP) (also called micro BTB) and complex two level predictors based on global history vectors (GShare or TAGE). The RAS has a repair mechanism on mispredicts resulting in a high prediction accuracy. The issue width of the execute pipeline is configurable. A distributed scheduler assigns micro operations to available execution units each containing some mix of functional units. Hereby one can select from eight different functional units. Similarly as with the Rocket core, the RoCC interface allows to add custom ISA extensions as accelerator implementations. The load-store unit is optimized for the superscalar out-of-order architecture. The data cache is organized into two dual-ported banks, which provides dual issuing and still allows an efficient 1R1W SRAM instantiation. FireSim can also be utilized as for Rocket.

There exists one documented tapeout called BROOM, which uses TSMC's 28 nm HPM process [18]. It has a built in 1 MB L2 cache and is designed to run at up to 1 GHz at 0.9 V, while its performance is specified with 3.77 CoreMark/MHz.

3.3 CVA6

CVA6 (formerly named Ariane) is an in order, single issue, 64-bit application class processor implementing the RV64GC standard [44]. The core is written in SystemVerilog and its micro-architecture is designed to reduce the critical path length while keeping IPC losses moderate.

CVA6 has a 6-stage pipeline, which can be compared to the 5-stage Rocket pipeline with an added stage for Program Counter (PC). The frontend contains a branch prediction with BTB, BHT, and RAS. Instructions are issued to six functional units within the execution stage: the ALU, a dedicated multiplier/divider, optional FPU (aimed to be IEEE 754-2008 compliants), CSR buffer, branch unit, and load/store unit (LSU). Timing critical components such as the register file and caches are designed with special care and can be configured for area or timing optimization.

The core has been integrated into both Chipyard and the OpenPiton[3] project, which simplifies the generation of a CVA6 based SoC, its simulation, and customization. Compared to a CVA6 core generated with OpenPiton, we observed a significant performance loss for the Chipyard generated variant. With the core and cache configurations selected within this work, the benchmark results drop by 68-83% when utilizing Chipyard. Primary reason for this is the intermediate TileLink translation required by Chipyard. The CVA6 core AXI interface connects to Chipyard's system bus (TileLink), which again connects to an AXI DDR4 interface. Hence, CVA6 has been evaluated in the following with the OpenPiton framework providing full performance.

CVA6 has been taped out six times in two different technologies, which is well documented [2]. The first tapeout named Poseidon is based on GlobalFoundries 22 nm FD-SOI technology (single core, 910 MHz, 0.8 V). Kosmodrom (1.3 GHz/300 MHz, 0.8 V) and Baikonur (1.0 GHz/250 MHz, 0.8 V) each evaluate performance- and power optimized variants of the CVA6 architecture and are again implemented in GlobalFoundries 22 nm FD-SOI technology. Scarabaeus (single core, 200 MHz, 1.2 V) and the most recent tapeout Urania (single core CVA6 and CV32E40P clusters, 100 MHz, 1.2 V) are based on the UMC 65 nm process.

3.4 SHAKTI C-Class

The SHAKTI Processor Program [26], initiated by the IIT Madras in 2014, focuses on developing power processors, SoCs, and peripheral IPs for an open-source ecosystem. So far, SHAKTI has released eight processors based on the open RISC-V ISA within three categories (base, multi-core, and experimental). SHAKTI C-Class, a member of the base family, is a controller grade processor designed for the IoT-, industrial-, and automotive segment. The core is designed for a frequency range from 500 MHz to 1.5 GHz and is capable to boot Linux and RTOS. In the following, the term SHAKTI refers to the SHAKTI C-Class processor.

[3] https://github.com/PrincetonUniversity/openpiton

The processor features an in-order 5-stage pipeline and supports both the RV32I and RV64I ISA. It is highly configurable, e.g., the S and M extensions can be selectively activated. The frontend contains a GShare two-level branch predictor and the execution stage is organized in three functional units (M-Box, F-Box, ALU). The core is also fully compatible with both AXI4 and TileLink interconnects.

The processors are written in Bluespec SystemVerilog (BSV), which can be transformed into synthesizeable Verilog code with an open-source compiler. Compared with other HDLs, BSV gives a higher level of abstraction to express structural and behavioral architectures. In contrast to the other three evaluated cores, SHAKTI is not integrated in the Chipyard framework. However, the SHAKTI project provides independent frameworks for creating SoC designs (shakti-soc), software development (shakti-sdk), and verification (e.g., RISC-V Trace Analyzer (RITA)). The ecosystem helps users to map the core on FPGA boards as well as to develop applications. This work utilizes the shakti-soc framework for an SHAKTI C-Class SoC implementation on an FPGA.

The processor has been fabricated in SCL 180 nm (RIMO, 350 MHz) and Intel 22 nm FinFET (RISECREEK, 70 MHz) technologies [12]. The performance of both tapeouts is specified with 1.68 DMIPS/MHz.

3.5 Summary of Analysis

Table 1 summarizes the characteristics of the Rocket, BOOM, CVA6, and SHAKTI processors. For traceability of our results, Table 1 also specifies the framework and core version (commit) utilized for further evaluation.

Table 1: Characteristics of different RISC-V implementations.

	Rocket	BOOM	CVA6	SHAKTI
Bits	32/64	64	64	32/64
Stages	5	10	6	5
Extensions	MAFDC	MAFDC	MAFDC	MAFDC
OoO exec	no	yes	no	no
Funct. Units	4	8	6	3
Inferfacing	TileLink	TileLink	AXI4	AXI4/TL
HDL	Chisel	Chisel	SV	BSV
License	BSD	BSD	SolderPad	BSD
Framework	Chipyard	Chipyard	OpenPiton	shakti-soc
Commit	1872f5d[1]	d77c2c3[2]	1793be6[3]	884fc43[4]

[1] https://github.com/chipsalliance/rocket-chip/
[2] https://github.com/riscv-boom/riscv-boom/
[3] https://github.com/openhwgroup/cva6/
[4] https://gitlab.com/shaktiproject/cores/c-class/

The different RISC-V implementations specify default architectural design parameters such as cache sizes and branch prediction buffer sizes. Performance, area, and power consumption results are affected by those predefined configurations. The following evaluation utilizes common architectural design settings being listed in Table 2. The hereby attained equal conditions provide a better comparability of the RISC-V cores. Arrays within the branch prediction unit (BHT, BTB, RAS) are generously dimensioned. The RISC-V

cores should operate close to their maximum possible processing performance when configured with the selected parameters. [44] shows that the IPC count already saturates for any RAS configuration larger than 2 and any BTB configuration larger than 8.

Other configuration options (e.g., pipeline registers for CVA6) are set to proposed default values. The OoO characteristic of the BOOM processor offers further configuration options which are not available for the other processors. Here we configured a medium to large sized variant with an issue width of 5.

Table 2: Common architectural design parameters utilized for the detailed comparison.

Parameter	Value
Branch History Table (BHT) depth	512
Branch Target Buffer (BTB) depth	32
Return Address Stack (RAS) depth	8
L1D cache size	16 KB
L1I cache size	16 KB

4 EVALUATION PLATFORMS

The RISC-V processors are compared for both FPGA deployment and ASIC synthesis, which addresses the differences of FPGA and ASIC implementations.

The FPGA tests have been performed on a VCU118 evaluation board containing an XCVU9P Xilinx Virtex UltraScale+ FPGA, which is manufactured in a 16 nm FinFET node. The device on the selected evaluation board possesses enough resources to implement all the cores and corresponding SoC designs in this state-of-the-art technology. Measurements are taken for the full SoC designs and standalone cores separately. All implementations have been run with the Xilinx Vivado Design Suite 2019.2.

For Rocket and BOOM there is no support to run a recent version (< 1 year old) of the RISC-V cores on the VCU118. Both the SiFive Freedom and lowRISC frameworks containing FPGA projects for Rocket and BOOM are not maintained anymore and already outdated. We developed a wrapper for the instantiation of any processor generated within the Chipyard framework on the VCU118. This allows to test current variants of Rocket, BOOM, and CVA6. As described in Sect. 3.3, OpenPiton is utilized instead of Chipyard for the CVA6 evaluation due to performance reasons.

To provide a fair comparison, default but identical settings are selected for FPGA synthesis and P&R for all processors. All power / area optimizations, e.g. within the shakti-soc framework, were carefully deactivated.

The ASIC comparisons are based on synthesis results of the GlobalFoundries 22FDX Fully-Depleted Silicon-On-Insulator (FD-SOI) technology. The planar process grows a ultra-thin transistor channel on top of a buried oxide insulator, which delivers FinFET-like performance and power efficiency. We used the INVECAS twelve track (12T) BASE standard cell library with a nominal voltage of 0.8 V. Memory macros have been generated with the INVECAS memory compiler. Single ported memory (S1P) is used preferentially and dual ported (R2PH) where required. Access schemes and tagging policies are specific to each RISC-V implementation and result in distinct array organizations, which required customized memory macros for each core despite of identical cache sizes.

Clock gating and medium power optimization efforts are activated, because it drastically reduces power consumption of the designs under evaluation but affects timing only marginally. The technology is not affected by temperature inversion for the selected synthesis parameters. Therefore, the operation condition assumes the worst corner with 0.72 V and 125 °C. The designs are synthesized with Cadence Genus Synthesis Solution Version 19.11-s087_1 using identical settings. The ASIC synthesis provides results for standalone RISC-V cores only, because no DDR IP has been available for the evaluation of complete SoC designs.

All RISC-V processors have been implemented in both technologies with best practice of FPGA / ASIC design development. However, no thorough optimizations of toolchain settings were analyzed. The RISC-V source code has been changed only where necessary (e.g., memory macro instantiation). Generally, no source code has been modified in order to improve the evaluation results (e.g., fixing critical paths). An exception is SHAKTI's cache design, because its very fine granular array instantiation heavily degrades performance, area, and power consumption results. We optimized the memory organization for the ASIC synthesis to countervail this to some degree; however, the cache design still represents a bottleneck within SHAKTI's architecture.

A team of designers could likely further optimize each RISC-V implementation by both fine-tuning the toolchain settings of the FPGA / ASIC design flow and more source code adaptions. Hereby, optimized results can be achieved for performance, area, and power consumption, but this should hold for all evaluated designs and therefore does not affect the general comparison.

5 DETAILED COMPARISONS

The detailed comparison of the application-class RISC-V cores is based on several evaluation criteria, whereas the first three extracted from implementation results of the FPGA deployment and ASIC synthesis. The *processing performance* is an important measure for the selection of a core for a project with specific computation requirements. The occupied *area* in silicon determines cost due to required FPGA or ASIC size. In particular battery-powered devices are afflicted with tight power constraints, hence the *energy efficiency* is another important criterion for processor selection.

5.1 Processing Performance Metrics

While Dhrystone [42] and CoreMark [5] are not well suited for evaluation of application-class and OoO cores, they are very common benchmarks and allow comparisons with smaller RISC-V variants. Hence, respective results will be provided. Additionally, with exception of CVA6 all RISC-V implementations will be stressed with the industry-standardized SPEC CPU 2017 [13] benchmark, which aims to compare compute intensive performance and covers a wide range of workloads.

Table 3 reports Dhrystone[4], CoreMark per MHz, and the harmonic IPC mean of the SPECintrate benchmarks. All values have

[4]Compiler settings are: `-DNO_PROTOTYPES=1 -DPREALLOCATE=1 -mcmodel=medany -static -std=gnu99 -O2 -ffast-math -fno-common -fno-builtin-printf -march=rv64imafd -mbranch-cost=2 -frename-registers`

been computed by execution of the benchmarks on the RISC-V SoC designs deployed on the FPGA evaluation board. Those benchmark results are technology agnostic.

Table 3: Benchmark and maximum frequency results of RISC-V implementations for the XCVU9P FPGA and the 22FDX ASIC technology.

Core	DMIPS per MHz	CoreMark per MHz	SPEC17 IPC	Fmax [MHz] XCVU9P	Fmax [MHz] 22FDX
Rocket	1.71	2.94	0.33	198	813
BOOM	3.87	6.25	0.50	88	943
CVA6	1.21	2.08	-	112	738
SHAKTI	1.70	2.84	0.23	136	685

Whereas Dhrystone and CoreMark are executed without restrictions on all four cores, several remarks apply for the SPEC17 benchmark. The VCU118 addressable DDR4 memory of 2x 2 GB is not sufficient for running the SPECintspeed suite requiring at least 12 GB memory. However, the utilized SPECintrate suite with more relaxed memory requirements yields similar results [31]. For the x264 benchmark test input data is provided; all others are executed with train input data. The SHAKTI design was only able to execute 5 out of the 10 benchmarks without faults. Due to Linux boot issues, it was not possible to run the SPEC benchmark for CVA6. Whereas Dhrystone and CoreMark on Rocket and BOOM do not benefit from an additional L2 cache in the selected configurations, it impacts the SPEC scores. Adding a 512 kB L2 cache improves the SPEC scores by 30.13% (Rocket) / 40.02% (BOOM). Detailed scores of the SPECintrate benchmark are given in Fig. 2.

The maximum processor core frequency is another performance factor and has been obtained by incrementally increasing the clock until timing violations were reported. These results are technology dependent and are specified for the Virtex UltraScale+ family and 22FDX ASIC synthesis separately. It is expected that the maximum frequency scales for each processor similarly when being deployed on another FPGA family or ASIC technology.

As expected for the only OoO-type processor under evaluation, BOOM leads the performance per MHz criterion and outpaces all others by more than a factor of 2. Rocket, SHAKTI, and CVA6 follow in the named order. The DMIPS/MHz measured by us coincides with the value reported for SHAKTI (1.72 in [15]) and falls short for CVA6 (1.65 in [44]). Reasons for this might be the use of different repository versions or differing compiler settings.

Regarding the maximum frequency, Rocket achieves the highest score for the FPGA implementation with 198 MHz and is second for the ASIC variant. BOOM is slowest of all four within the FPGA and fastest of all four within the ASIC. For this high discrepancy between maximum frequencies two root causes have been identified. 1) BOOM is the only design that spreads over two Super Logic Regions (SLRs) within the Virtex device requiring a segmentation for the FPGA implementation. 2) BOOM very rigorously instantiates arrays which can be translated into memory macros and allow an efficient ASIC implementation.

Rocket's and SHAKTI's maximum ASIC frequency is limited by the data cache latency. The suboptimal memory organization within SHAKTI hereby results in a 15% slower design compared to Rocket. The critical path of the BOOM synthesis contains PTW logic. CVA6 clocking is restricted by L1D logic.

The product of both criteria, (i) benchmark results per MHz and (ii) maximum frequency accounts for the overall processor performance, which is being depicted in Fig. 3. BOOM by far leads this performance comparison for the ASIC technology, but is similar (Dhrystone and CoreMark) or inferior (SPEC) to Rocket for the FPGA technology due to its frequency limitation. Hence, the OoO BOOM would be the first choice for a high-performance ASIC implementation, but it cannot fully outperform the in-order Rocket when being deployed on an FPGA. The CVA6 and SHAKTI implementation only achieve 40 to 84% of the performance of Rocket, depending on the benchmark and technology.

5.2 Area Metrics

Both the Vivado and Genus toolchains generate detailed resource utilization reports, which facilitates an area comparison. The results apply for the Xilinx Virtex UltraScale+ architecture and 22FDX node respectively, but it is expected that they scale for other FPGA families and ASIC technologies. The results are reported for designs that have been generated with a relaxed clock constraint (50 MHz for XCVU9P FPGA and 500 MHz for 22FDX ASIC). Only a moderate resource and area increase has been observed for respective Fmax clock constraints.

Fig. 4 depicts the SoC resource utilization results of the four evaluated RISC-V projects implemented on the XCVU9P FPGA. The SoC contains, in addition to the RISC-V core itself, further processor components such as a memory interface and peripherals. For a more detailed discussion of the RISC-V core area, its resources are marked hatched.

5.2.1 Core Area (FPGA). The comparison of RISC-V core sizes (not counting further SoC resources) emphasizes differences of the evaluated implementations. The core size reflects the complexity of the processor architecture and has to be considered n-times for a multi-core design with n cores. The core contains resources for its pipeline, the frontend with L1I cache and branch prediction, the L1D cache, and PTW. Rocket has the lowest resource utilization for all resource types, with exception of BRAM. The complexity of BOOM's OoO pipeline results in a very high LUT, register, and DSP utilization. SHAKTI implements the caches based on single ported sub-arrays with a depth of 64 entries and a width of 64 bit. This filigree segmentation results in an inefficient BRAM resource utilization on the FPGA; compared to Rocket it requires 3.2 times as many BRAM resources. Compared to CVA6, SHAKTI has similar register and DSP utilization, but it requires 38% more LUTs. The more complex GShare branch predictor of SHAKTI is one reason for this. Furthermore, we observe that SHAKTI's L1D cache structure instantiates disproportionately many LUTs.

5.2.2 SoC Area (FPGA). The area of the remaining SoC structures adds to the core area and is determined by the utilized framework. It reflects the complexity of other processor components and contains resources for clocking, the memory interface, external core devices (Boot ROM, interrupt controllers, debug unit), and peripherals (UART, JTAG, SPI). The evaluation board provides DDR4 memory

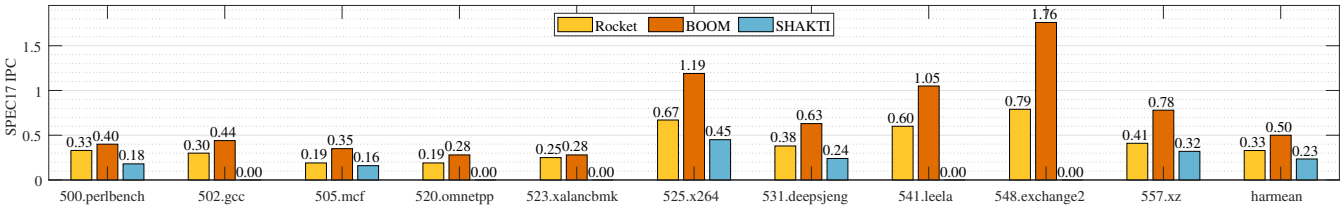

Figure 2: SPECintrate IPC scores.

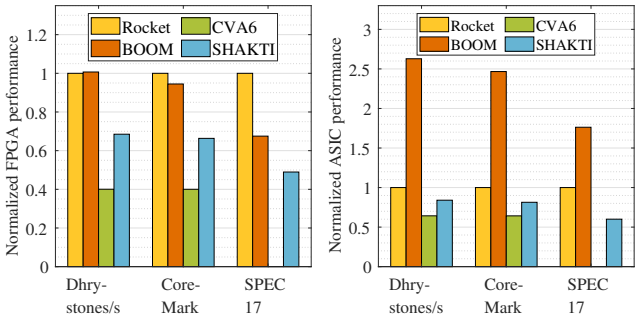

Figure 3: RISC-V performance normalized to BOOM. Left: for XCVU9P FPGA. Right: for 22FDX ASIC synthesis. Higher is better.

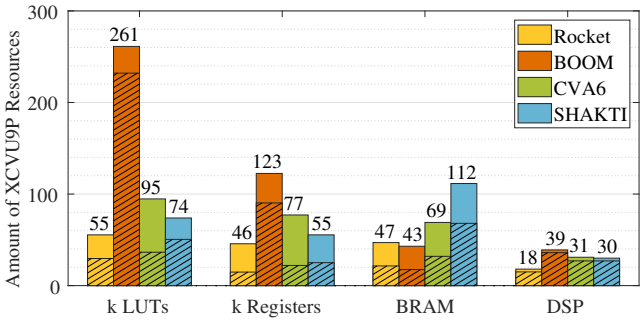

Figure 4: Resource utilization of RISC-V SoC implementations on Xilinx Virtex UltraScale+ XCVU9P FPGA. RISC-V core resources are marked hatched. Lower is better.

and all four evaluated implementations instantiate a therefor required Xilinx DDR4 controller IP [43], which contributes the largest resource demand within the remaining SoC structures. The SoC structures utilize very similar amounts of FPGA resources (non-hatched area of Fig. 4). This is as expected, because the frameworks all instantiate the same peripherals, albeit in different variants. Only OpenPiton stands out with an increased resource utilization, which is because of provisioned structures for a multi-core design.

5.2.3 Core Area (ASIC). The ASIC synthesis is performed for the RISC-V cores only (without SoC resources). Fig. 5 plots the RISC-V core area over delay constraints and shows only a moderate area increase for tight timing constraints. The very left marker of each line denotes its area for the respective core's maximum frequency. The area converges for relaxed clocking to 0.22 mm2 (Rocket), 0.92 mm2 (BOOM), 0.52 mm2 (CVA6), and 0.84 mm2 (SHAKTI). The Rocket core has the smallest footprint and could fit more than four times into the complex BOOM. CVA6 has a medium core area, but SHAKTI's footprint is relatively large due to inefficient memory macro instantiation.

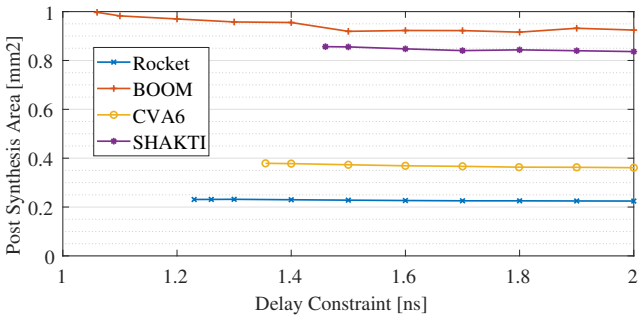

Figure 5: Post-synthesis 22FDX area-delay curves for RISC-V cores.

5.3 Power Metrics

5.3.1 Power Consumption. The Power Management Bus (PMBus) has been used to measure both the static *power consumption* of the VCU118 evaluation board and the dynamic power consumption of respective RISC-V implementations. All measurements are performed for 0.85 V V_{CCINT}, a die temperature of 27.0 °C, and a RISC-V clock frequency of 50 MHz; results are averaged from of 128 measuring points. The static power consumption of the VCU118 (FPGA device plus DDR4, RLDRAM, and Flash) has been determined by loading an empty design into the FPGA device. It is defined by the utilized FPGA evaluation board and independent of the RISC-V core; hence, identical values are reported in Table 4.

After loading the respective RISC-V design into the FPGA, the power consumption has been measured again while executing the Dhrystone benchmark. The increased consumption compared to the empty FPGA design with no clock input is a measure for dynamic power consumption; results are provided in Table 4.

The Genus synthesis reports are evaluated for respective ASIC power consumptions under relaxed clock constraints (500 MHz). Hereby a switching activity based on a Dhrystone simulation has been provided. Note that in contrast to the FPGA power consumption results, Table 4 specifies the ASIC power consumption for the standalone RISC-V core only.

SHAKTI proves to be the most power efficient core when being deployed on the FPGA device. The fine granular distribution

Table 4: Power consumption [mW] on XCVU9P FPGA (measured on line) and for 22FDX ASIC synthesis (estimated by synthesis tool).

Core	FPGA SoC static	FPGA SoC dynamic	ASIC Core static	ASIC Core dynamic	FPGA SoC MOp/J
Rocket	3080	1820	4.14	15.76	17.4
BOOM	3080	3030	26.37	139.03	31.7
CVA6	3080	1995	9.27	26.30	11.9
SHAKTI	3080	1660	24.20	23.81	17.5

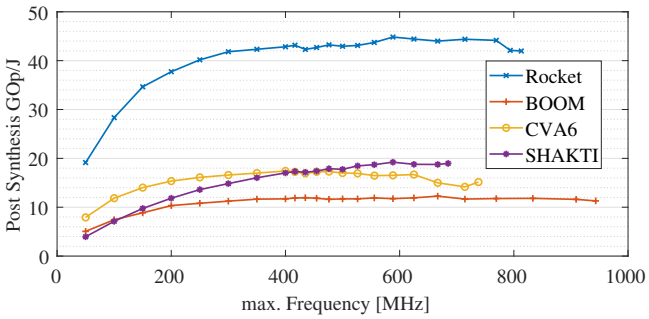

Figure 6: Energy efficiency of RISC-V cores for the 22FDX technology.

of memory to BRAM is not detrimental to dynamic FPGA power consumption. With a similar IPC as Rocket, it is also very energy efficient. Rocket shows a 2% and CVA6 a 12% higher dynamic power consumption compared to SHAKTI. The dynamic power consumption of BOOM is almost twice as large as of the most efficient in-order variant.

The FPGA results do not correlate to ASIC power consumption. As the memory macros contribute a major part of the static power consumption, SHAKTI's high score can be explained by its microarchitecture allowing memory macros of only shallow depths.

5.3.2 Energy Efficiency. The so far discussed power consumption metric is reported for a fixed operating frequency only. Furthermore, it penalizes large but powerful processors. High IPC scores typically require high complexity resulting in high power consumption. Comparing the *energy efficiency* overcomes this problem, because it measures the completed workload in relation to consumed energy. The energy efficiency is listed in Table 4 for an operation on the FPGA device with 50 MHz as Mega Operations per Joule. The large static power proportion of the Virtex FPGA makes RISC-V cores with a low IPC inefficient; hence, BOOM has the highest energy efficiency in this technology. For the 22FDX technology, Fig. 6 gives more detailed energy efficiency information and plots the GOp/J results of all four evaluated RISC-V cores over frequency. At lower frequencies (left of Fig. 6), the static power consumption dominates, resulting in a decreased energy efficiency. When getting close to the maximum achievable frequency, the synthesis tool optimizes the design for performance. This is traded for power consumption, e.g., by instantiating a larger proportion of SLV cells. The resulting decrease of energy efficiency for very high frequencies can be observed particularly for Rocket and CVA6.

Rocket achieves the highest maximum GOp/J score (40.4), followed by SHAKTI (32.5), CVA6 (20.2), and BOOM (12.3). Rocket is 3.6 times more efficient than BOOM, which illustrates how BOOM traded high performance for energy efficiency. SHAKTI's memory macros contribute to a relatively high static power consumption (comp. Table 4), being the reason for an more distinctive energy efficiency degradation at low to medium frequencies. Above 500 MHz SHAKTI and CVA6 reach comparable energy efficiencies; however, both are only half as efficient as Rocket.

5.4 Summary of Comparisons

The Rocket implementation achieves high scores for all evaluation criteria, except for ASIC processing performance. It features a high FPGA performance in combination with lowest FPGA resource utilization, smallest ASIC footprint, and high energy efficiency. Many configuration options simplify its adoption for a wide range of academic and commercial projects. BOOM, the only OoO core analyzed within this work, can replace the Rocket core. It is best in class for ASIC performance, but this is traded for a high FPGA resource utilization, ASIC area footprint, and low energy efficiency. SHAKTI is most power and energy efficient when deployed on an FPGA. However, its L1 cache aspect ratio has a negative impact for the ASIC design in particular. It limits the maximum frequency and results in a large memory area and power consumption. Once this issue is fixed, its performance, area utilization, and energy efficiency can achieve more optimized results.

6 CONCLUSION

This work compared the four open-source application-class RISC-V processor implementations Rocket, BOOM, CVA6, and SHAKTI C-Class. The fair comparison is based upon common configuration settings and execution of equal benchmarks on identical platforms.

The results show big differences regarding processing performance (up to 3.1x), area, resource utilization, power consumption, and energy efficiency (up to 3.6x). The Rocket core achieved best scores for many criteria, but the other implementations also have their strengths. E.g., BOOM achieves the highest ASIC processing performance, SHAKTI is best in class for FPGA energy efficiency.

The large variations of results highlight the importance of processor selection. The data provided in this work helps to make a good choice for future projects with varying processing needs. There is clearly no optimal implementation in general. The ranking order depends on the selected technology (FPGA / ASIC) and primary requirements (performance / cost / efficiency).

This paper only presents a snapshot in time, because all RISC-V projects are actively enhanced by many contributors and the results discussed here vary by each version. Furthermore, only a specific configuration has been analyzed for each RISC-V implementation. Future work will analyze the effect of those two additional dimensions on the evaluation scores.

ACKNOWLEDGMENTS

This work is part of BMBF FKZ 16ES1003 "KI-PRO".

REFERENCES

[1] 2020. AndesCore Processors. http://www.andestech.com/en/products-solutions/andescore-processors/. Accessed: 2020-11-23.

[2] 2020. Chips by IIS. http://asic.ethz.ch/. Accessed: 2020-11-18.

[3] 2020. CloudBEAR processors for the widest range of applications. https://cloudbear.ru/products.html. Accessed: 2020-11-23.

[4] 2020. Codasip RISC-V Processors. https://codasip.com/risc-v-processors/. Accessed: 2020-11-23.

[5] 2020. CoreMark - An EEMBC Benchmark. https://www.eembc.org/coremark/. Accessed: 2020-11-23.

[6] 2020. Lizard Core. https://github.com/cornell-brg/lizard. Accessed: 2020-11-23.

[7] 2020. mor1kx - an OpenRISC processor IP core. https://github.com/openrisc/mor1kx. Accessed: 2020-11-23.

[8] 2020. PicoRV32 - A Size-Optimized RISC-V CPU. https://github.com/cliffordwolf/picorv32. Accessed: 2020-11-23.

[9] 2020. PolarFire SoC. https://www.microsemi.com/product-directory/soc-fpgas/5498-polarfire-soc-fpga. Accessed: 2020-11-23.

[10] 2020. RISC-V Cores and SoC Overview. https://github.com/riscv/riscv-cores-list. Accessed: 2020-11-23.

[11] 2020. SCR5 efficient application core (RV32 or RV64). https://syntacore.com/page/products/processor-ip/scr5. Accessed: 2020-11-23.

[12] 2020. SHAKTI. https://shakti.org.in/tapeout.html. Accessed: 2020-11-23.

[13] 2020. SPEC CPU 2017. https://www.spec.org/cpu2017/. Accessed: 2020-11-23.

[14] 2020. *SpinalHDL, VexRiscv: A FPGA friendly 32 bit RISC-V CPU implementation.* https://github.com/SpinalHDL/VexRiscv Accessed: 2020-11-23.

[15] 2021. SHAKTI C-Class Read the Docs. https://c-class.readthedocs.io/en/latest/benchmarking.html. Accessed: 2021-02-04.

[16] Krste Asanović, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Ben Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David A. Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. 2016. *The Rocket Chip Generator.* Technical Report UCB/EECS-2016-17. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

[17] Rahul R. Balwaik, Shailja R. Nayak, and Amutha Jeyakumar. 2013. Open-Source 32-Bit RISC Soft-Core Processors. *IOSR Journal of VLSI and Signal Processing* 2 (2013), 43–46.

[18] C. Celio, P. Chiu, K. Asanović, B. Nikolić, and D. Patterson. 2019. BROOM: An Open-Source Out-of-Order Processor With Resilient Low-Voltage Operation in 28-nm CMOS. *IEEE Micro* 39, 2 (March 2019), 52–60. https://doi.org/10.1109/MM.2019.2897782

[19] Christopher Celio, Pi-Feng Chiu, Borivoje Nikolic, David A. Patterson, and Krste Asanović. 2017. *BOOM v2: an open-source out-of-order RISC-V core.* Technical Report UCB/EECS-2017-157. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-157.html

[20] C. Chen, X. Xiang, C. Liu, Y. Shang, R. Guo, D. Liu, Y. Lu, Z. Hao, J. Luo, Z. Chen, C. Li, Y. Pu, J. Meng, X. Yan, Y. Xie, and X. Qi. 2020. Xuantie-910: A Commercial Multi-Core 12-Stage Pipeline Out-of-Order 64-bit High Performance RISC-V Processor with Vector Extension : Industrial Product. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA).* 52–64. https://doi.org/10.1109/ISCA45697.2020.00016

[21] Niket K. Choudhary, Salil V. Wadhavkar, Tanmay A. Shah, Hiran Mayukh, Jayneel Gandhi, Brandon H. Dwiel, Sandeep Navada, Hashem H. Najaf-abadi, and Eric Rotenberg. 2011. FabScalar: Composing Synthesizable RTL Designs of Arbitrary Cores within a Canonical Superscalar Template. In *Proceedings of the 38th Annual International Symposium on Computer Architecture* (San Jose, California, USA) *(ISCA '11).* Association for Computing Machinery, New York, NY, USA, 11–22. https://doi.org/10.1145/2000064.2000067

[22] Rangeen Basu Roy Chowdhury, Anil K. Kannepalli, Sungkwan Ku, and Eric Rotenberg. 2016. AnyCore: A synthesizable RTL model for exploring and fabricating adaptive superscalar cores. *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (2016), 214–224.

[23] P. Davide Schiavone, F. Conti, D. Rossi, M. Gautschi, A. Pullini, E. Flamand, and L. Benini. 2017. Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS).* 1–8. https://doi.org/10.1109/PATMOS.2017.8106976

[24] Stefano Di Mascio, Alessandra Menicucci, Eberhard Gill, Gianluca Furano, and Claudio Monteleone. 2019. Leveraging the Openness and Modularity of RISC-V in Space. *Journal of Aerospace Information Systems* 16, 11 (2019), 454–472. https://doi.org/10.2514/1.I010735

[25] Alexander Dörflinger, Yejun Guan, Sören Michalik, Sönke Michalik, Jamin Naghmouchi, and Harald Michalik. 2020. ECC Memory for Fault Tolerant RISC-V Processors. In *Architecture of Computing Systems – ARCS 2020*, André Brinkmann, Wolfgang Karl, Stefan Lankes, Sven Tomforde, Thilo Pionteck, and Carsten Trinitis (Eds.). Springer International Publishing, Cham, 44–55.

[26] Neel Gala, Arjun Menon, Rahul Bodduna, GS Madhusudan, and V Kamakoti. 2016. SHAKTI processors: An open-source hardware initiative. In *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID).* IEEE, 7–8.

[27] N. Gala, A. Menon, R. Bodduna, G. S. Madhusudan, and V. Kamakoti. 2016. SHAKTI Processors: An Open-Source Hardware Initiative. In *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID).* 7–8. https://doi.org/10.1109/VLSID.2016.130

[28] C. Heinz, Y. Lavan, J. Hofmann, and A. Koch. 2019. A Catalog and In-Hardware Evaluation of Open-Source Drop-In Compatible RISC-V Softcore Processors. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig).* 1–8. https://doi.org/10.1109/ReConFig48160.2019.8994796

[29] R. Höller, D. Haselberger, D. Ballek, P. Rössler, M. Krapfenbauer, and M. Linauer. 2019. Open-Source RISC-V Processor IP Cores for FPGAs — Overview and Evaluation. In *2019 8th Mediterranean Conference on Embedded Computing (MECO).* 1–6. https://doi.org/10.1109/MECO.2019.8760205

[30] Jens Korinth, Jaco Hofmann, Carsten Heinz, and Andreas Koch. 2019. The TaPaSCo Open-Source Toolflow for the Automated Composition of Task-Based Parallel Reconfigurable Computing Systems. In *Applied Reconfigurable Computing*, Christian Hochberger, Brent Nelson, Andreas Koch, Roger Woods, and Pedro Diniz (Eds.). Springer International Publishing, Cham, 214–229.

[31] A. Limaye and T. Adegbija. 2018. A Workload Characterization of the SPEC CPU2017 Benchmark Suite. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS).* 149–158. https://doi.org/10.1109/ISPASS.2018.00028

[32] M. Makni, M. Baklouti, S. Niar, M. W. Jmal, and M. Abid. 2016. A comparison and performance evaluation of FPGA soft-cores for embedded multi-core systems. In *2016 11th International Design Test Symposium (IDT).* 154–159. https://doi.org/10.1109/IDT.2016.7843032

[33] S. Mashimo, A. Fujita, R. Matsuo, S. Akaki, A. Fukuda, T. Koizumi, J. Kadomoto, H. Irie, M. Goshima, K. Inoue, and R. Shioya. 2019. An Open Source FPGA-Optimized Out-of-Order RISC-V Soft Processor. In *2019 International Conference on Field-Programmable Technology (ICFPT).* 63–71. https://doi.org/10.1109/ICFPT47387.2019.00016

[34] E. Matthews, Z. Aguila, and L. Shannon. 2018. Evaluating the Performance Efficiency of a Soft-Processor, Variable-Length, Parallel-Execution-Unit Architecture for FPGAs Using the RISC-V ISA. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM).* 1–8. https://doi.org/10.1109/FCCM.2018.00010

[35] Junya Miura, Hiromu Miyazaki, and Kenji Kise. 2020. A portable and Linux capable RISC-V computer system in Verilog HDL. arXiv:2002.03576 [cs.AR]

[36] D. Richmond, M. Barrow, and R. Kastner. 2018. Everyone's a Critic: A Tool for Exploring RISC-V Projects. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL).* 260–2604. https://doi.org/10.1109/FPL.2018.00052

[37] Rui Jia, Colin Yu Lin, Zhenhong Guo, Rui Chen, Fei Wang, Tongqiang Gao, and Haigang Yang. 2014. A survey of open source processors for FPGAs. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL).* 1–6. https://doi.org/10.1109/FPL.2014.6927482

[38] SEMICO Research Corporation 2019. *RISC-V Market Analysis The New Kid on the Block* (cc315-19 ed.). SEMICO Research Corporation.

[39] SiFive, Inc. 2019. *SiFive U54 Manual* (v19.08p0 ed.). SiFive, Inc.

[40] Andrew Waterman and Krste Asanović. 2017. *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10.* Technical Report. EECS Department, University of California, Berkeley. https://content.riscv.org/wp-content/uploads/2017/05/riscv-privileged-v1.10.pdf

[41] Andrew Waterman, Yunsup Lee, David A. Patterson, and Krste Asanović. 2016. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.1.* Technical Report UCB/EECS-2016-118. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-118.html

[42] Reinhold P. Weicker. 1984. Dhrystone: A Synthetic Systems Programming Benchmark. *Commun. ACM* 27, 10 (Oct. 1984), 1013–1030. https://doi.org/10.1145/358274.358283

[43] Xilinx Inc. 2019. *UltraScale Architecture-Based FPGAs Memory IP, PG150* (v1.4 ed.). Xilinx Inc.

[44] F. Zaruba and L. Benini. 2019. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 11 (Nov 2019), 2629–2640. https://doi.org/10.1109/TVLSI.2019.2926114

[45] S. Zhang, A. Wright, T. Bourgeat, and A. Arvind. 2018. Composable Building Blocks to Open up Processor Design. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).* 68–81. https://doi.org/10.1109/MICRO.2018.00015

[46] Jerry Zhao, Abraham Gonzalez, Ben Korpan, and Krste Asanovic. 2020. SonicBOOM: The 3rd Generation Berkeley Out-of-Order Machine. In *Fourth Workshop on Computer Architecture Research with RISC-V (CARRV 2020).*