# Open-Source RISC-V Processor IP Cores for FPGAs – Overview and Evaluation

Roland Höller, Dominic Haselberger,
Dominik Ballek, Peter Rössler
Department of Electronic Engineering
University of Applied Sciences Technikum Wien
Höchstädtplatz 6, 1200 Vienna, Austria
{hoeller, roessler}@technikum-wien.at

Markus Krapfenbauer, Martin Linauer
Kapsch TrafficCom AG
Am Europlatz 2, 1120 Vienna, Austria

*Abstract*—**Advances in semiconductor miniaturization are an important driver for Field Programmable Gate Arrays (FPGAs) since their invention in the 1980s. The increasing number of available on chip logic resources on one hand and on the other hand a decrease in part costs let the FPGA market grow steadily in recent years. It comes thus at no surprise that more and more microprocessors are integrated into programmable logic devices as they represent the central functionality in many digital systems. In parallel to these technological developments the open-source hardware community grew steadily in the last two decades. More than hundred open-source CPU cores can thus be found and selecting a core for a design project has to be done with care. In this work we thus want to focus on open-source 32-bit CPU IP cores suitable for FPGAs and which support the upcoming free and open RISC-V instruction set architecture that has some interesting advantages when compared to commercial CPU cores (as will be outlined in the paper). An overview on available projects and activities will be given and evaluation results for a selection of cores will be presented.**

***Keywords-Field Programmable Gate Arrays; Programmable System-on-Chip; Open-Source CPU Cores; RISC-V; IP Core***

## I. INTRODUCTION

As the number of available logic resources in modern FPGA (Field Programmable Gate Array) devices increases as steadily as the achievable maximum clock frequency of synchronous digital designs in such technologies, use of programmable logic becomes available in many different electronic products. To fill the devices with useful logic in accordance to the design specification in the typically tight time frames of digital design projects, the use of IP (Intellectual Property) cores is a proven and valuable methodology since many years. The design IP, in the usual case coming from different vendors, will be integrated with the design project's RTL (Register Transfer Level) code to form designs as complex as SoCs (System-on-Chips). Such SoCs contain interfaces to external devices like Ethernet connectivity as well as static or dynamic memories and often integrate one or even several CPUs (Central Processing Units) capable of executing programs ranging from rather simple control applications up to complex operating systems.

The various IP cores large FPGA designs are made up of can come in different forms: As a technology dependent netlist, as RTL code together with a license and in the typical case with guarantees about the capabilities and the quality of the IP as well as costs. In recent years, however, the idea of open-source hardware design started to become as popular as the idea has already been for years in the software domain [13]. There, as everyone knows, the source code of application programs or even operating systems is publicly available. It is important to note that open-source is not to be mistaken for free: Although code may be available open-source it might still be licensed for non-commercial use only, to give an example.

There are commercial CPU IP cores available from different vendors, which often are optimized for a certain technology and thus may achieve better performance [2]. However, these commercial cores also come with some limitations that have to be considered when designing an electronic system using such IP:

- The existing commercial CPU cores from FPGA vendors that are provided in form of a netlist are typically technology-dependent. Thus, porting the design to devices from other FPGA vendors is not possible without changing the CPU core as well.
- There also exist FPGA families with hard-wired CPU cores (e.g., the ARM Cortex architecture is very popular in recent devices). Although these CPU cores provide an excellent performance (by supporting clock frequencies above 1 GHz) parts costs for such FPGA families are often higher when compared to smaller FPGA devices which can make the hard IP CPU cores unsuitable for cost-sensitive applications.
- Connecting external memory to the previously mentioned FPGA devices with hard IP CPU cores is typically a must since the amount of available FPGA-internal block RAM is often far too small in order to hold startup and application code for the CPU. In other words, the system is no longer a single-chip solution which again induces higher costs for an electronic product.

Due to these limitations, commercial CPU cores are out of scope of our overview and evaluation. Open-source hardware designs can be found on the Internet on GitHub, the OpenCores website, or the CERN Open Hardware Repository and this is

This work was financially supported by the Department MA23 of the City of Vienna in context to the project "FPGA 4.0", project number MA23 19-07, as well as the Austrian Federal Ministry for Digital and Economic Affairs (BM:DW) and the National Foundation for Research, Technology and Development as related to the Josef Ressel Center "Innovative Platforms for Electronic-Based Systems" (INES), managed by the Christian Doppler Research Association.
978-1-7281-1740-9/19/$31.00 ©2019 IEEE

where the CPU IP cores subject to this work have been found[1]. There are three relatively up to date lists on that very topic, which are updated from time to time and worth visiting[2].

Another important thing when it comes to IP cores for microprocessors is the supported ISA (Instruction Set Architecture). It links the hardware and the software world since it specifies for a software compiler how to best map the high-level application software's constructs into low level instructions that the CPU will then execute [7]. In this context various ISAs have been invented over the last decades, most of them proprietary, but some of them free [1,6]. Amongst the free ISAs is the relatively young RISC-V (Reduced Instruction Set Computer) instruction set, which is now maintained by the RISC-V Foundation [6,7,8].

The RISC-V ISA is free, open, and modular [7,8]. It claims to avoid incremental development and changes as opposed to proprietary ISAs. Its design should be cost-effective, simple, operate on small code sizes, isolate architecture from implementation, and leave room for growth [8]. The RISC-V ISA has already seen several implementations as well as real silicon (i.e. ASIC – Application Specific Integrated Circuit) and FPGA realizations. This is reflected when looking at the number of scientific publications that make reference to the RISC-V ISA (see Fig. 1). When looking at the last five years, a full text search for significant keywords shows a decline in the number of annually added publications which make a reference to a commercial soft CPU IP core. In the same time frame the number of publications making reference to an open-source CPU IP core increased, which is especially the case for RISC-V cores and a motivation for this paper.

When it comes to make use of a CPU IP core for real life projects other than pure performance evaluations, the core will have to be connected with peripheral modules and extensions such as general purpose I/Os, timers, universal asynchronous receive transmit (UART), memory controllers, serial peripheral interface (SPI), and the like. To interface to such additional IP cores in a seamless way the CPU core is best offering a standard on-chip bus interface for connection. We thus treated an on-chip bus interface as a must have requirement for our overview of cores in Section III.

The contributions of this work are twofold: first an overview of open-source CPU IP cores is given and important properties are presented; second three selected RISC-V CPU IP cores are implemented and evaluated with (as the first step of our investigations) a focus on hardware-related issues. The data about open-source RISC-V CPU IP cores presented in this paper is as of March 2019. The mention of this date is important as many of the online projects evolve at a rapid pace but at the same time unfortunately often lack concise administration of releases.

[1] See the websites at https://github.com, https://opencores.org, and https://ohwr.org respectively for open-source CPU and other IP cores.

[2] See https://riscv.org/risc-v-cores/ or http://parallel.princeton.edu/openpiton/open_source_processors.php or https://opencores.org/projects/up_core_list/summary.

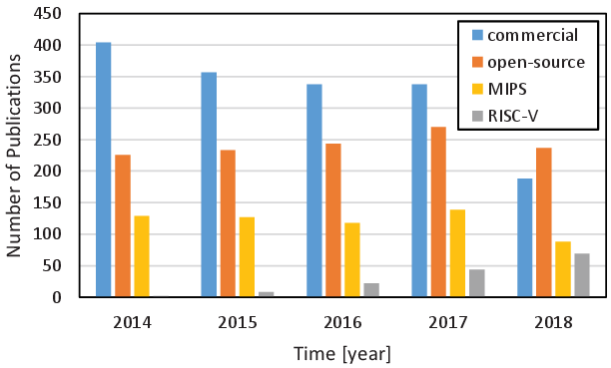


Figure 1. Comparison of the annually added number of scientific publications including keywords concerning selected soft processor IP cores for FPGAs in the IEEEXplore digital library. The commercial category includes hits for ARM Cortex-M1, Cast BA22-DE, Intel NIOS II and Xilinx microblaze. The open-source category includes RISC-V, MIPS, openrisc, LEON3, OpenSparc and Mico32 cores. For comparison hits for RISC-V as well as MIPS are shown additionally. Growing interest in RISC-V cores for FPGAs is clearly visible.

## II. Related Work and State of the Art

The basic idea of open-source CPU cores is of course not new. It comes thus as no surprise that tables, comparisons, and evaluations of open-source CPU cores have already been published in scientific circles. However, to the best of our knowledge a structured overview of properties of open-source CPU IP cores focusing on the RISC-V architecture and a reproducible evaluation of selected RISC-V cores as presented herein is new.

In [1] Rui Jia et al. give a survey in which they claim to have analyzed 178 open-source processors. In their selection process the properties "stability" and "completion status of the design project" are mentioned as basis to narrow down the high number of potential candidates to 68 processors. The details of this selection process, or a clear definition of the properties "stability" and "completeness" is unfortunately not disclosed by the authors. Their further analysis documents how the number of processors is again reduced to finally seven by taking the license, the ISA, and properties like "synchronous or asynchronous design" as well as "single- or multicore" into account. The subsequent evaluation and comparison of the seven remaining cores is detailed and also compares them to commercial cores. Therefore we have taken the four cores that fit into our requirements into our Table I for comparison. Because of the year of publication of [1] RISC-V cores there have not been considered at all.

In [2] Makni et al. compare commercial as well as open-source soft processor cores as candidates to build an embedded multi-core system. By virtue of detailed evaluations they conclude that the achievable clock frequency of processor cores for FPGAs is typical proportional to the number of pipeline stages and that commercial soft cores typically outperform open-source ones because of their optimization towards the target technology. RISC-V cores have not been taken into account by [2].

In [3] Balkind et al. set out to establish an open-source manycore research framework. Although manycore design is out of the scope of our work, Balkind et al. compare the basic properties of their framework with other open-source processor design projects. Their list contains 22 open-source processor cores of which four are also mentioned in our Table I. The RISC-V ISA is present in [3] with the RISC-V Rocket and RISC-V Boom processor cores from UC Berkeley Architecture Research. One implementation of the Rocket core, called freedom, is present also in our work in its 32-bit version. But no other RISC-V cores are mentioned by Balkind et al. in [3].

In [4] six open-source processor cores are surveyed by Balwaik et al. Three of the six are taken for comparison also in this work. However, [4] does not make reference to any RISC-V cores and it is furthermore not clear if the discussed cores were actually implemented by the authors.

Tong et al. present a survey of soft core processors for embedded systems [5]. Three commercial soft cores and two open-source cores are compared, but because of the year of publication of [5] no reference to RISC-V cores was made. It is also unclear if the implementation results do come from the author's own implementation runs or are just taken from other publications.

In [16] Matthews et al. evaluate five different RISC-V soft processors for their performance and efficiency. It is shown that the variable-latency, parallel-execution architecture of the Taiga soft processor core is superior in performance to the fixed pipeline architecture of four other cores. In [21] two cores implementing the SH-2 ISA are compared to the SCR1 RISC-V core and finally Richmond et al. present in [18] an open-source tool set for exploring RISC-V projects.

As already mentioned in Section I the RISC-V RV32 instruction set has already seen implementations in ASIC or FPGA technologies documented in scientific publications like [16] for ORCA, [19] for RI5CY, [20] for zero-riscy, [21] for SCR1, [22] for Hummingbird, [23] for Shakti, [16] for PicoRV32, and [15,16,17] for Taiga. In that context it will be interesting to follow, if the announced MIPS Open Initiative to make the Release 6 of the MIPS ISA free and open in Q1 2019 will give a similar push to open-source hardware as RISC-V obviously did.

By studying the implementation results in [1,2,3,16] and taking our own results of Section IV, it can be deduced that pipelined RISC processor cores today can achieve operating frequencies in modern FPGA technologies roughly between 20 MHz and 250 MHz. This is a factor of six to ten slower than an ASIC implementation [3]. Thereby the cores use between zero and nine pipeline stages for executing the instructions.

As already mentioned in Section I, commercial cores are out of the scope of this work because of our focus on tool- and technology independence as well as our requirement of full control over the core's source code. Absence of license fees and openness of the instruction set architecture are other arguments for our focus on RISC-V processor cores.

One of our goals was to list only open-source cores, which offer a standard on-chip bus interface. This is in order to connect to peripheral and other maybe custom cores and do useful work in their design, see Section III for a full list of our requirements. It comes as no surprise thus, that some of the listed CPU IP cores exist within generators or platforms that are capable of coping with the on-chip bus interface, size of caches, handle hook-up of peripheral units, offer debug support, or even support multi-core architectures. Often software support or even ports of operating systems are included or existing as well. Examples of such larger projects or complete systems are the PULP (Parallel Ultra Low Power) platform including PULPino [10], OpenPiton [3], the Rocket Chip Generator [24], MiSoC with OpenRISC, and the LEON3 processor with the GRLIB [25].

Please note that the mentioned topics of open-source SoC platforms or generators, as well as low level driver support for such SoCs, and support for operating systems are regarded as important and crucial to bring a system as complex as an SoC to live.

TABLE I. COMPARISON OF SELECTED OPEN-SOURCE PROCESSOR IP CORES. ONLY OPEN-SOURCE CORES WITH FREE ISA, 32 BIT DATA WIDTH, AVAILABLE STANDARD BUS INTERFACE, SYNCHRONOUS DESIGN, TOOL- AND TECHNOLOGY INDEPENDENT CODE BASE, AND EXISTING COMPILER SUPPORT WERE TAKEN INTO ACCOUNT HERE. THE FIRST FOUR CORES ARE LISTED FOR COMPARISON ONLY. [SOURCE: [1,2,3,4,5] AND OWN RESEARCH.]

| Name of processor | Instruct. Set Architecture and Data Width | No.of Pipeline Stages | Bus Architecture | MMU | FPU | HDL | Compiler | Debug Support | License | Last Update | Multi-Core | In Order | Cache | JTAG | Peripherals Included |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Amber | ARM v2a, 32 bit | 5 | WB | N | N | Verilog | GCC | Y | LGPL | 2017 | N | Y | Y | N | Y |
| Lattice Mico 32 | LatticeMico32, 32 bit | 6 | WB | N | N | Verilog | GCC | Y | GPL | 2017 | N | Y | Y | Y | Y |
| openrisc | ORBIS, 32 bit | 5 | WB | Y | Y | Verilog | GCC | Y | LGPL | 2019 | N | Y | Y | N | Y |
| Leon3 | SPARC V8, 32 bit | 7 | AHB | Y | Y | VHDL | GCC | Y | GPL | 2018 | Y | Y | Y | Y | Y |
| freedom | RISC-V, 32 bit | 5 | TL/AXI | N | Y | Chisel | GCC | Y | BSD | 2018 | N | Y | Y | Y | Y |
| ORCA | RISC-V, 32 bit | 5 | WB/AXI | N | N | VHDL | GCC | N | BSD | 2019 | N | Y | Y | N | N |
| RI5CY | RISC-V, 32 bit | 4 | AXI | N | Y | Verilog | GCC | Y | Solderpad | 2018 | N | Y | N | Y | Y |
| zero-riscy | RISC-V, 32 bit | 2 | AXI | N | N | Verilog | GCC | Y | Solderpad | 2018 | N | Y | N | N | Y |
| OPenV | RISC-V, 32 bit | 3 | AXI | N | N | Verilog | GCC | N | MIT | 2018 | N | Y | N | N | Y |
| VexRiscv | RISC-V, 32 bit | 5 | AXI | Y | N | SpinalHDL | GCC | Y | MIT | 2019 | N | Y | Y | Y | Y |
| Roa Logic RV12 | RISC-V, 32 bit | 6 | AHB/WB | N | N | Verilog | GCC | Y | Non | 2018 | N | Y | Y | N | N |
| SCR1 | RISC-V, 32 bit | 4 | AXI | N | N | Verilog | GCC | Y | Solderpad | 2019 | N | Y | N | Y | N |
| Hummingbird E200 | RISC-V, 32 bit | 2 | AXI | N | N | Verilog | GCC | Y | Apache | 2019 | N | Y | Y | Y | Y |
| Shakti | RISC-V, 32 bit | 3 | AXI | N | N | Bluespec | GCC | N | BSD | 2019 | N | Y | Y | N | Y |
| ReonV | RISC-V, 32 bit | 7 | AHB | Y | Y | VHDL | GCC | Y | GPL v3 | 2018 | Y | Y | Y | Y | Y |
| PicoRV32 | RISC-V, 32 bit | 0 | AXI | N | N | Verilog | GCC | N | ISC | 2018 | N | Y | N | N | Y |
| SweRV EH1 | RISC-V, 32 bit | 9 | AXI | N | N | Verilog | GCC | Y | Apache | 2019 | N | Y | Y | Y | N |
| Taiga | RISC-V, 32 bit | 3± | AXI | Y | N | Verilog | GCC | N | Apache | 2018 | N | Y | Y | N | N |
| potato | RISC-V, 32 bit | 5 | WB | N | N | VHDL | GCC | N | BSD | 2018 | N | Y | Y | N | Y |

And they are in the author's opinion especially important if it is the case of open-source hardware. In this paper, however, the focus is on CPU IP cores only and evaluation or comparison of open-source SoC generators or platforms and software topics of such systems is out of scope, but subject to possible future research [13].

## III. Overview and Comparison of RISC-V Cores

As already outlined in the previous sections the number of open-source CPU IP cores that can be found is quite high and a structured selection is to be based on a clear set of requirements. Within the course of two funded research projects we were searching for CPU IP that was apt to base a prototype system on and that would accompany us for the next couple of years to do research on several topics in relation to FPGA design (such as hardware/software co-design or investigations towards hardware security and functional safety). We summarized our requirements for the selection of a CPU IP core thus as follows:

- The instruction set architecture should be free. No license issues should arise even if our design would later be used by ourselves or a research partner for commercial designs.
- The IP core should be open-source. We want to be able to fix bugs or modify the design.
- We decided to limit ourselves to an address space and a data width of 32 bit. This would reduce complexity but still provide enough headroom for our planned research activities in the foreseeable future.
- The support of a standard on-chip bus interface of said data width was a very important requirement. Many different modules of various proveniences will be connected to the design by research staff as well as students, the complexity thereof ranging from simple UARTs to DMA (Direct Memory Access) capable Ethernet MACs and the like.
- The circuitry of the IP core should adhere to a strict synchronous design methodology. It should at best use single edge clocking and have a system wide reset signal.
- Although our primary target technology is FPGAs, the source code should be technology independent as much as possible. On one hand we want to be able to port the design to other FPGA technologies and on the other hand keep the option to implement the design into an ASIC.
- The source code should be tool independent, i.e. not rely on the usage of a certain design tool be it a commercial EDA (Electronic Design Automation) tool or an open-source one.
- Finally, the core and ISA should be supported by a standard C/C++ compiler, which is actively developed and maintained.

The open-source CPU IP cores listed in Table I fulfill the just mentioned requirements. The table starts with the name of the processor IP and then lists the instruction set architecture the core is compatible to together with the width of the on-chip data bus. The number of pipeline stages in the processor instruction engine is listed and the standard of the supported bus architecture, where other IP cores can be hooked up. The support of MMU (Memory Management Unit) is of relevance when an operating system like Linux is to be run on the design. When a FPU (Floating Point Unit) can be implemented with the core it is indicated in the corresponding column. All of the cores are supported by the GCC compiler and some even have debug support. Only the freedom core, however, is supported by OpenOCD via JTAG. Verilog and SystemVerilog are the dominant languages used to implement the cores and from our experience use of VHDL, Verilog, or SystemVerilog is the best choice. Only a few cores support to build multi-core systems and all cores execute in order.

## IV. Evaluation and Results

Out of our list of sixteen open-source 32-bit RISC-V IP cores (Table I) we have selected three cores for closer evaluation and implementation (other cores will be evaluated in the future, see Section V). We started our investigations with these three cores mainly because of the completeness of their designs with on-chip busses and peripherals. In the first step of our evaluations that are presented herein the focus is on hardware-related issues such as FPGA resource consumption. Both a CPU core with small footprint (such as the PicoRV32 core) as well as a core with a relatively high amount of resources (like the Pulpino RI5CY) have been selected for the first investigations. The target technology for our evaluations was a Xilinx 7-series Artix XC7A35T FPGA mounted on an ARTY evaluation board.

### A. Use Case A – Freedom Core

The freedom core implements the RISC-V RV32IMAC instruction set. The core is derived from the open-source rocket-chip generator [12]. The Verilog source code here is generated out of a high-level hardware description in Chisel. It is a single-issue CPU with a five stage pipeline. In the implementation we evaluated, the blocks of the system around the CPU are interconnected using the TileLink system bus, but a bridge towards the AXI bus does exist. The design is synchronous to its single clock domain and uses a synchronous high active reset. The debug module is elaborate and is supported by the openOCD and Eclipse development environment. Peripherals are attached via a separate peripheral bus and support UART, PWM (Pulse Width Modulation), SPI, I2C, as well as QSPI, see Fig. 2.

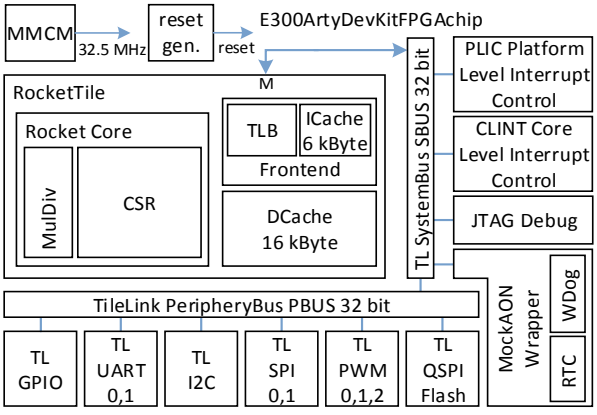


Figure 2. Block diagram of freedom RISC-V CPU IP core based design.

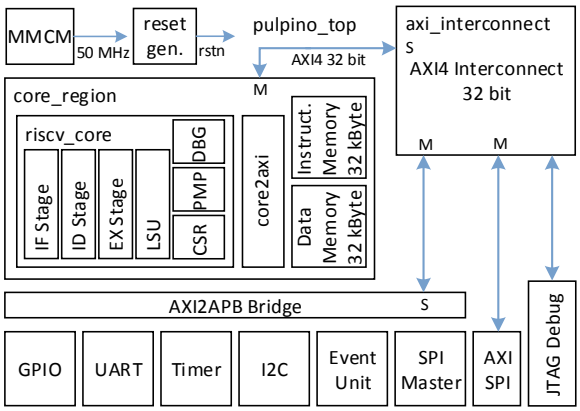


Figure 3. Block diagram of RISCY RISC-V CPU IP core based design.

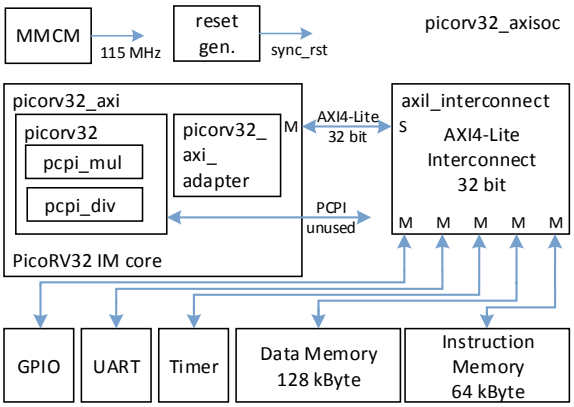


Figure 4. Block diagram of PicoRV32 RISC-V CPU IP core based design.

### B. *Use Case B – Pulpino RI5CY*

The RI5CY is an open-source 32-bit RISC-V core written in SystemVerilog [14]. It supports the RV32IMC instruction set. The core is part of the PULP family of platforms, maintained by the University of Bologna and the ETH Zürich [10]. RI5CY uses a single-issue four stage instruction pipeline and does not use caches. It is designed for separate instruction and data memories. Although the RI5CY repository comes with an implementation for another FPGA board, we ported the design to use our standard evaluation board to allow for comparison of implementation figures. The design is synchronous using the rising clock edge of a single clock domain and implements a low active asynchronous reset for all storage elements. Some peripherals are attached to the processor core via a 32-bit AXI on-chip bus system in order to provide interfaces to GPIO, UART, I2C, SPI, etc. (see Fig. 3) and represent a platform called PULPino [11]. Minor changes have been made to operate the design stand alone, where program code is copied to internal RAM from external SPI flash via a bootloader at startup.

### C. *Use Case C – PicoRV32*

PicoRV32 is a small RISC-V ISA compliant open-source CPU IP core intended for simple embedded applications. It supports the RV32IMC set of instructions, where only the RV32IM has been used in the course of this work. Adding to the core's simplicity pipelining is not implemented and caches for instruction or data do not exist. Execution of instructions thus takes between 3 and 6 clock cycles for most instructions, multiply, divide, and shift take up to 40 cycles. The core uses a single memory interface for instructions and data where an access to instruction memory is indicated by a control signal. The core's repository, however, offers an adapter to connect to a standard AXI4-Lite bus system, which has been made use of in our evaluation design.

The design is synchronous using the rising clock edge of a single clock domain and implements a low active synchronous reset for all storage elements. This core has a small area footprint and runs at a relatively high clock rate in FPGA technologies. We found, however, that the reset signal in the Verilog code is also connected to design logic which is a bad design practice especially when targeting ASIC technologies.

Not being able to operate the reset alone without interfering with design logic would prohibit straight forward test insertion for production test in, e.g., a standard cell technology. The core has a separate PCPI (Pico Co-Processor Interface) where the built-in multiplier and division units are connected to. Here peripherals are attached to the processor core via a 32-bit AXI Lite on-chip bus system in order to provide interfaces to GPIO, UART, and a Timer unit, see Fig. 4. Program code is implemented in the on-chip FPGA 64 kByte BRAM instruction memory. The software running on the presented system was compiled and linked with GCC 8.2.0. The resulting contents of the 64 kByte instruction memory is then included as preloaded memory content via the Xilinx Vivado 2018.2 design software.

### D. *Evaluation Results*

Table II presents the resource utilization of the three RISC-V based designs in terms of Look-Up Tables (LUTs), Flip-Flops (FFs), Block RAMs (BRAMs), Digital Signal Processing Blocks (DSPs), Mixed-Mode Clock Managers (MMCMs) and number of used IO pins (IOs) as well as the clock frequency of the designs (the numbers in brackets point to the resources of the CPU core only). The Freedom and the Pulpino RI5CY based designs show a similar resource consumption while the PicoRV32 proves the very small footprint as mentioned in Sub-Section IV-C. The number of used BRAMs depends on the needs of the software application and can be configured at design time. All three CPU systems (including the peripherals like GPIO, UART, Timer Units …) have been successfully tested on the FPGA board by running some demo applications. Furthermore, the performance has been evaluated in terms of Dhrystone MIPS (Million Instructions Per Second) per MHz CPU clock. Here, the PicoRV32 shows a lower performance than its counterparts due to the missing pipeline stage and shared memory bus.

TABLE II. COMPARISON OF RISC-V BASED USE CASES A, B AND C.

| Design | LUTs | FFs | BRAMs | DSPs | MMCMs | I/Os | $f_{CLK}$ [MHz] | DMIPS per MHz |
|---|---|---|---|---|---|---|---|---|
| **Freedom** | 13062 (2692) | 7662 (1311) | 6 | 2 | 1 | 57 | 32.5 | 1.61 |
| **Pulpino RI5CY** | 14616 (6748) | 8959 (2577) | 16 | 8 | 1 | 20 | 50 | 1.10 |
| **PicoRV32** | 2123 (1765) | 1509 (1078) | 48 | - | 1 | 19 | 115 | 0.13 |
| **Artix XC7A35T** | 33280 | 41600 | 50 | 90 | 5 | 250 | | |

## V. CONCLUSIONS AND FUTURE WORK

The free and modular RISC-V ISA fosters implementations of IP cores making most often use of the easy to use base instruction set and (due to its open source nature) provides some advantages over commercial CPU cores (Section I). When comparing such IPs and selecting amongst a number of available options, the project's requirements and the core's different properties come into play. In this work an overview of existing open-source RISC-V CPU IP cores has been given (Section II and III) and three cores were implemented and evaluated in more detail (Section IV).

The focus on our evaluations presented herein is on hardware-related issues. In the future we plan to continue our research and evaluation by investigation of additional CPU cores listed in Table I. We plan to further evaluate the CPU cores with a benchmark like MiBench [9] to assess CPU performance in more detail. The next steps will also deal with JTAG debug support using OpenOCD, use of 64-bit CPU cores, multi-core support, incorporation of a memory management unit, and operating system support. Furthermore the authors are searching for an open-source solution for a tool that automatically handles the system bus interconnections between CPU core and peripherals, before using the built SoC for performance evaluations or power optimizations.

## REFERENCES


[1] Rui Jia et al., "A survey of open source processors for FPGAs", 2014 24th International Conference on Field Programmable Logic and Applications (FPL), Munich, 2014, pp. 1-6.

[2] M. Makni, M. Baklouti, S. Niar, M. W. Jmal and M. Abid, "A comparison and performance evaluation of FPGA soft-cores for embedded multi-core systems", 2016 11th International Design & Test Symposium (IDT), Hammamet, 2016, pp. 154-159.

[3] J. Balkind, M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, A. Lavrov, M. Shahrad, A. Fuchs, S. Payne, X. Liang, M. Matl, and D. Wentzlaff, "OpenPiton: An Open Source Manycore Research Framework", SIGOPS Oper. Syst. Rev. 50, 2, March 2016, pp. 217-232.

[4] R. R. Balwaik et al., "Open-source 32-bit risc soft-core processors", IOSR Journal of VLSI and Signal Processing, 2(4), 2013, 43-46.

[5] J. G. Tong, I. D. L. Anderson and M. A. S. Khalid, "Soft-Core Processors for Embedded Systems," 2006 International Conference on Microelectronics, Dhahran, 2006, pp. 170-173.

[6] D. A. Patterson, and A. Waterman, "The RISC-V Reader: An Open Architecture Atlas", Strawberry Canyon, 2017.

[7] D. A. Patterson & J. L. Hennessy, "Computer Organization and Design: The Hardware/Software Interface RISC-V Edition Cambridge, 2018.

[8] K. Asanović, and D. A. Patterson, "Instruction sets should be free: The case for risc-v", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, 2014.

[9] MiBench: A free, commercially representative embedded benchmark suite by Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, Richard B. Brown, IEEE 4th Annual Workshop on Workload Characterization, Austin, TX, December 2001.

[10] D. Rossi et al., "PULP: A parallel ultra low power platform for next generation IoT applications," 2015 IEEE Hot Chips 27 Symposium (HCS), Cupertino, CA, 2015, pp. 1-39.

[11] M. Gautschi et al., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," IEEE Transactions on Very Large Scale Integration Systems, vol. 25, no. 10, pp. 2700-2713, Oct. 2017.

[12] K. Asanovic, R. Avizienis, J. Bachrach et al., "The rocket chip generator", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016.

[13] G. Gupta, T. Nowatzki, V. Gangadhar and K. Sankaralingam, "Kickstarting Semiconductor Innovation with Open Source Hardware," in *Computer*, vol. 50, no. 6, pp. 50-59, 2017.

[14] M. Gautschi *et al*., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 25, no. 10, pp. 2700-2713, Oct. 2017.

[15] E. Matthews and L. Shannon, "TAIGA: A new RISC-V soft-processor framework enabling high performance CPU architectural features," *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Ghent, 2017, pp. 1-4.

[16] E. Matthews, Z. Aguila and L. Shannon, "Evaluating the Performance Efficiency of a Soft-Processor, Variable-Length, Parallel-Execution-Unit Architecture for FPGAs Using the RISC-V ISA," *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Boulder, CO, 2018, pp. 1-8.

[17] J. R. G. Ordaz and D. Koch, "A Soft Dual-Processor System with a Partially Run-Time Reconfigurable Shared 128-Bit SIMD Engine," *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Milan, 2018, pp. 1-8.

[18] D. Richmond, M. Barrow and R. Kastner, "Everyone's a Critic: A Tool for Exploring RISC-V Projects," *2018 28th Int. Conf. on Field Progr. Logic and Applications (FPL)*, Dublin, 2018, pp. 260-2604.

[19] C. Palmiero, G. Di Guglielmo et al., "Design and Implementation of a Dynamic Information Flow Tracking Architecture to Secure a RISC-V Core for IoT Applications," *2018 IEEE High Performance extreme Computing Conference (HPEC)*, Waltham, MA, 2018, pp. 1-7.

[20] P. Davide Schiavone *et al*., "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation*, Thessaloniki, 2017, pp. 1-8.

[21] K. Hagiwara *et al*., "A two-stage-pipeline CPU of SH-2 architecture implemented on FPGA and SoC for IoT, edge AI and robotic applications," *2018 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, Yokohama, 2018, pp. 1-3.

[22] S. Huang and R. Chen, "FPGA-Based IoT Sensor HUB," *2018 International Conference on Sensor Networks and Signal Processing (SNSP)*, Xi'an, China, 2018, pp. 139-144.

[23] S. Gupta, N. Gala, G. S. Madhusudan and V. Kamakoti, "SHAKTI-F: A Fault Tolerant Microprocessor Architecture," *2015 IEEE 24th Asian Test Symposium (ATS)*, Mumbai, 2015, pp. 163-168.

[24] Y. Lee *et al*., "An Agile Approach to Building RISC-V Microprocessors," *IEEE Micro*, vol. 36, no. 2, pp. 8-20, Mar.-Apr. 2016.

[25] A. Kchaou, W. El Hadj Youssef and R. Tourki, "Performance evaluation of multicore LEON3 processor," *2015 World Symposium on Computer Networks and Information Security*, Hammamet, 2015, pp. 1-4.


## APPENDIX – INTERNET LINKS

| Name of Core | Internet Link to the Core's Source Repository |
|---|---|
| Amber | http://opencores.org/project,amber |
| Lattice Mico 32 | http://www.latticesemi.com/en/Products/DesignSoftwareAndIP/IntellectualProperty/IPCore/IPCores02/LatticeMico32.aspx |
| openrisc | http://openrisc.io/ |
| Leon3 | http://www.gaisler.com/index.php/products/processors/leon3 |
| freedom | https://github.com/sifive/freedom |
| ORCA | https://github.com/vectorblox/orca |
| RI5CY | https://github.com/pulp-platform/riscv |
| zero-riscy | https://github.com/pulp-platform/zero-riscy |
| OPenV | https://github.com/onchipuis/mriscv |
| VexRiscv | https://github.com/SpinalHDL/VexRiscv |
| Roa Logic RV12 | https://github.com/roalogic/RV12 |
| SCR1 | https://github.com/syntacore/scr1 |
| Hummingbird E200 | https://github.com/SI-RISCV/e200_opensource |
| Shakti | https://gitlab.com/shaktiproject |
| ReonV | https://github.com/lcbcFoo/ReonV |
| PicoRV32 | https://github.com/cliffordwolf/picorv32 |
| SweRV EH1 | https://github.com/westerndigitalcorporation/swerv_eh1 |
| Taiga | https://gitlab.com/sfu-rcl/Taiga |
| potato | https://github.com/skordal/potato |