INTERNATIONAL JOURNAL OF INTEGRATED ENGINEERING
ISSN: 2229-838X e-ISSN: 2600-7916

IJIE

Vol. 16 No. 1 (2024) 312-317
https://publisher.uthm.edu.my/ojs/index.php/ijie

Performance Evaluation of RISC-V Microcontroller System on FPGA: A Study of the NEORV32 Core

Chung Wei Peng[1], Asral Bahari Jambek[1,2]*, Sreedharan Baskara Dass[3], Lim Teik Wah[4], Lalin L. Laudis[5], Avinash Yadlapati[4], Gnaneshwara Chary Udari[6], B. Rama Sanjeeva Reddy[6]

[1] *Faculty of Electronic Engineering & Technology, Universiti Malaysia Perlis, Arau, 02600, Perlis, MALAYSIA*
[2] *Centre of Excellence for Micro System Technology (MiCTEC), Universiti Malaysia Perlis, Arau, 02600, Perlis, MALAYSIA*
[3] *Intelligent Circuit Engineering Sdn Bhd, Puchong, Selangor, MALAYSIA*
[4] *Intel Technology Sdn Bhd, Bayan Lepas, Penang, MALAYSIA*
[5] *Mar Ephraem College of Engineering and Technology, Malankara Hills, Elavuvilai, Marthandam, Tamil Nadu, INDIA*
[6] *B V Raju Institute of Technology, Narsapur, INDIA*

*Corresponding Author: asral@unimap.edu.my
DOI: https://doi.org/10.30880/ijie.2024.16.01.026

Article Info

Received: 14 December 2023
Accepted: 23 February 2024
Available online: 2 June 2024

Keywords

RISC-V, NEORV32, CoreMark, hardware utilization, FPGA

Abstract

This paper evaluates a RISC-V microcontroller system on an FPGA platform using the NEORV32 core. This research aims to identify performance gaps in the NEORV32 system on an FPGA. The evaluation was carried out using the CoreMark benchmark programs. The hardware utilisation of the NEORV32 core is examined using Quartus Prime software with a particular focus on slice look-up tables (LUTs), total registers, memory bits, RAM blocks, and DSP blocks. In this work, two NEORV32 implementations are evaluated, which are RV32I and RV32I with M and Zfinx extension (RV32I_MC_zfinx). The effect of dedicated hardware and special operations on the performance of the processors is also evaluated on an FPGA board. The experiment results show that RV32I_MC_zfinx consumes 55% and 65% more LUT and registers resources, respectively, compared to the RV32I. Implementing hardware accelerators to RV32I_MC_zfinx results in a 48% increase in CoreMark score. Compared with other existing RISC-V cores, NEORV32 is a good option for embedded system development since it balances performance and resource efficiency for low-power applications.

1. Introduction

The Instruction Set Architecture (ISA) is the foundation of computer architecture that defines the interface between hardware and software and governing tasks that the CPU can execute [1]. At the processor level, there are many implementations of ISA. Each standard adheres to its parent ISA while offering some customisation [2]. RISC-V is one of the popular choices among emerging ISAs, where it is based on Reduced Instruction Set Computing (RISC) that prioritizes processor efficiency over instruction length.

RISC-V was introduced at the University of California, Berkeley, in 2010. It is an open-source ISA compared to other closed-source ISAs such as ARM and x86 [3]. Because of RISC-V openness, it has gained widespread adoption and support, encouraging collaborative work by various communities such as research, education, and industry players [4]. Because of its adaptability and support, RISC-V has emerged as a good choice for various applications, particularly in the Internet of Things (IoT) domain [3].

This paper aims to evaluate a RISC-V microcontroller system based on the NEORV32 core [11]. Based on the RV32I instruction set, the NEORV32 core provides a flexible and customisable RISC-V processor system-on-chip (SoC). It implements two-stage pipeline architecture for faster instruction processing and higher clock frequencies [5]. It uses an integrated memory system that includes programmable instruction and data caches to improve data throughput. Furthermore, NEORV32 includes diverse peripheral modules and interfaces, such as UART, SPI, GPIO, timers and counters, I2C, and PWM. This allows seamless integration with external devices and expands the system's capabilities [5]. In this work, several NEORV32 systems are implemented on an FPGA platform, and their performance and hardware utilisation are analysed.

The rest of the paper is organised as follows. Sections 2 discuss the existing work on RISC-V and NEORV32. The experimental methodology is discussed in section 3, while section 4 highlights our experimental result. Finally, section 5 concludes the paper.

2. Literature Review

Various literatures have studied different aspects of RISC-V cores and their performance. This section discusses the relevant literature, highlighting significant discoveries from prior research. Holler et al. [5] analysed FPGA-compatible open-source 32-bit CPU IP cores that support the RISC-V instruction set architecture. They compared RISC-V cores based on clock frequency, area, power consumption, and performance characteristics. This study offers valuable insights into the performance of various RISC-V cores, especially for those who need to select suitable cores for specific applications. In [6], Dörflinger et al. conducted a similar comparative survey of RISC-V cores. They performed performance benchmarking on both FPGA and ASIC platforms. The differences between RISC-V core implementations and their potential suitability for different application domains were examined in this literature. Heinz et al. [7] compared various open-source RISC-V softcore processors using. Embedded processor benchmarks such as Dhrystone, Embench, and CoreMark were used to evaluate these processors for various hardware platforms. This assessment compares various aspects of processor performance, such as integer arithmetic, memory access, and control flow.

Low-power applications are another focus of RISC-V core research. Bora and Paily [8] discussed a RISC-V-based microarchitecture optimised for low-power applications. Their research introduced branch prediction, out-of-order execution, and an advanced pipeline to improve performance and reduce power consumption. From the experimental result, the Dhrystone and CoreMark benchmarks show their proposed method outperforms the traditional RISC-V cores architecture, making it ideal for power-efficient applications.

Other than hardware, software support for RISC-V has been extensively researched. Mezger et al. [9] provided an overview of the current software support for RISC-V instruction set architectures. These authors covered various RISC-V software development tools, operating systems, and libraries. Developers can write code in high-level languages, debug applications, and simulate RISC-V implementations using extensive software support. These include GCC, LLVM/Clang compilers, GDB, OpenOCD debuggers, and various simulators.

While the existing work compares various RISC-V implementations, this paper investigates the performance and hardware utilisation of the NEORV32 core on an FPGA platform. This work aims to determine NEORV32's suitability for embedded system development. NEORV32 is a customisable RISC-V processor for system-on-chip (SoC) designed to meet the wide range of embedded system development requirements [11]. The NEORV32 core, based on the RISC-V RV32I base instruction set with a 2-stage pipeline design that balances between efficient instruction processing and higher clock frequencies. This results in improved overall performance.

The NEORV32 unified memory system is a key feature of the NEORV32 system, which includes instruction and data caches. These caches are easily customisable in size, allowing developers to tailor the system to specific application requirements. The NEORV32 architecture is enhanced by various peripheral modules and interfaces such as UART, SPI, GPIO, timers/counters, I2C, and PWM. With a wide range of interfaces, the NEORV32-based system's capabilities can be expanded through seamless integration with external devices. Cache sizes, peripheral modules, and memory mapping can be customised to meet specific application's need. In general, the NEORV32 architecture provides many features and customisation options, making it an appealing choice for a wide range of embedded system applications. Figure 1 shows the NEORV32 SoC architecture, highlighting its components and interconnections. Figure 2 depicts the pipelined multi-cycle architecture in detail.

3. Methodology

Several customised versions of the NEORV32 core were evaluated in this study. Each version was optimised for specific criteria to evaluate the impact of different configurations on performance and hardware utilisation. Table 1 shows the full list of tested NEORV32 core versions and their features.

Figure 3 shows the FPGA implementation process flow perform in this work. The first step in the process is to prepare the application programme for execution on the NEORV32 processor. To accomplish this, a toolchain is used to compile the source code and generate executable, processor-compatible files. The RISC-V GCC toolchains designed for Linux operating systems were chosen as the toolchain for this project. These toolchains provide RISC-V architecture support for compiling and debugging an application code.

Once the application program is ready, the NEORV32 processor is implemented on the FPGA using the Quartus Prime 181 Lite software. Then, the application program is uploaded from the host computer into the processor using UART communication. To monitor and analyse the output of programme execution by NEORV32 processor, GTKTerm, a serial terminal application, is used. This real-time monitoring of program execution aids in verifying the application's correctness. For optimal performance, the serial terminal settings are configured according to the following parameters: baud Rate (19200), data bits (8), stop bit (1), parity bits (None), and transmission/flow control protocol (none).

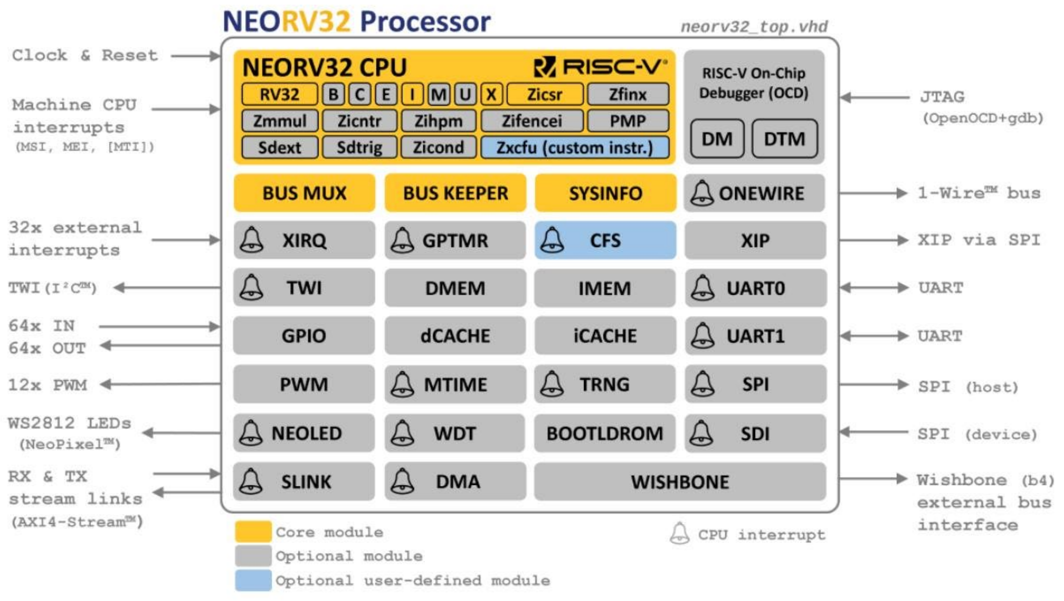

Fig. 1 *Overview of the NEORV32 SoC*

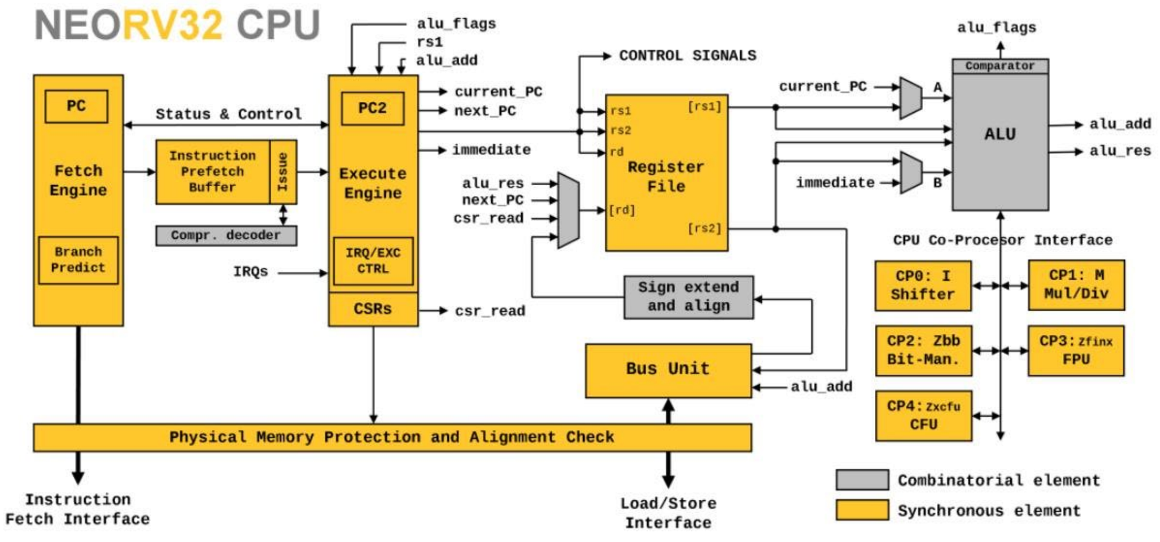

Fig. 2 *NEORV32 pipelined multi-cycle architecture*

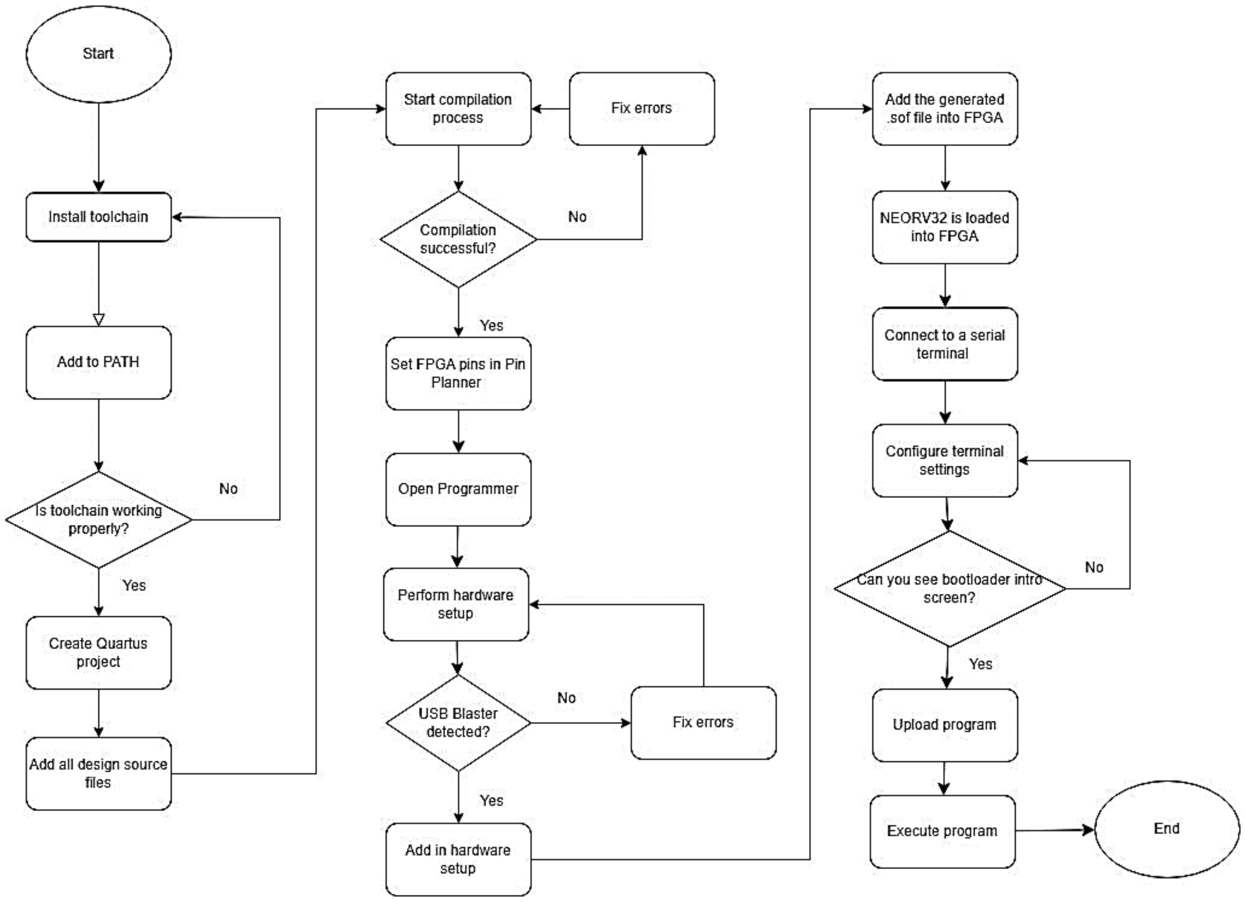
Start
Install toolchain
Add to PATH
Is toolchain working properly?
No
Yes
Create Quartus project
Add all design source files
Start compilation process
Fix errors
Compilation successful?
No
Yes
Set FPGA pins in Pin Planner
Open Programmer
Perform hardware setup
USB Blaster detected?
No
Fix errors
Yes
Add in hardware setup
Add the generated .sof file into FPGA
NEORV32 is loaded into FPGA
Connect to a serial terminal
Configure terminal settings
Can you see bootloader intro screen?
No
Yes
Upload program
Execute program
End

Fig. 3 *Implementation of FPGA process flow*

4. Result and Discussion

The evaluation results of the NEORV32 core versions and their corresponding optimisations are presented in this section. Hardware utilisation metrics, CoreMark benchmark scores, and a comparison of NEORV32 hardware utilisation on various FPGA platforms are discussed.

The hardware utilisation results of various NEORV32 core configurations on the DE1-SoC FPGA platform are shown in Table 2. Adding the Zfinx extension to the RV32I configuration significantly increased slice look-up tables (LUTs) and total registers by about 55% and 65%, respectively. This increase can be attributed to the additional integer register file required for floating-point data operations in the Zfinx extension feature. When the area was optimised by disabling the C-extension, LUTs decreased by 12.5% while total registers increased by 3.3%.
Memory usage was also assessed using internal instruction memory (IMEM) and internal data memory (DMEM). As expected, all configurations had high memory bit usage due to using a significant portion of the IMEM (32kB) and DMEM (16kB). RV32IMC zfinx +Perf and RV32IMC zfinx +Perf +Area configurations have additional instruction caches, which increase total memory bits.

Table 3 shows the results of the total RAM and DSP blocks required for each configuration. The total blocks for both RAM and DSP increased as more features were enabled in specific configurations since these features required more dedicated hardware resources.

The NEORV32 core's maximum operating frequency (Fmax) was important in determining at which maximum frequency the processor can operate reliably. However, as the complexity of NEORV32 increased with the addition of new features, the critical path of the design was impacted, resulting in a decrease in Fmax. During the evaluation process, this trade-off between increasing performance and optimising areas was considered.

The CoreMark benchmark results for each NEORV32 configuration are shown in Table 4. CoreMark scores represent the performance of various configurations. When the M and Zfinx extensions were activated, the average CPI (Cycles Per Instruction) increased because RV32IM zfinx necessitated more cycles for intricate multiplication operations. However, the CoreMark score rose, indicating an overall improvement in performance. Enabling performance options such as FAST MUL brought down the average CPI by utilising DSP slices for multiplication, resulting in a higher CoreMark score by 48%. When the area was optimised, the CoreMark score decreased, highlighting the trade-off between performance and area. Additionally, disabling the C-extension raised the CPI as it required more cycles for non-compressed instructions.

Table 5 compares the NEORV32 core to other existing RISC-V cores. NEORV32 utilised the highest number of LUTs among the tested cores while remaining within an acceptable range for low-power applications. Taiga achieved the highest CoreMark score, while NEORV32 demonstrated competitive performance, falling in the middle of the tested cores. This comparison showcases NEORV32 as a flexible and viable choice for embedded system development, offering an excellent balance of performance and resource efficiency. The subsequent sections will examine the implications of these findings and their relevance to the study's objectives.

Table 1 *Tested NEORV32 cores*

Tested Cores	Description
RV32I	32-bit RISC-V core with the Base Integer extension support
RV32IMC_zfinx	Addition of M-extension and Zfinx-extension
RV32IMC_zfinx + Perf	Addition of mapping of complex CPU operations to dedicated hardware and using DSP slices for multiplication + 2kB caches
RV32IM_zfinx + Perf + Area	Disabling of C-extension and map CPU shift operation to shifter unit

Table 2 *Hardware utilization results (LUTs, total registers, block memory bits***)**

Configuration	LUTs	Total Registers	Block Memory Bits
RV32I	1,148	1363	428,032
RV32IMC_zfinx	1,779	2245	428,032
RV32IMC_zfinx + Perf	2,827	2770	465,352
RV32IM +Perf +Area	2,476	2866	465,352

Table 3 *Hardware utilization results (DSP blocks, RAM blocks, Fmax)*

Configuration	DSP Blocks	RAM Blocks	Fmax (MHz)
RV32I	0	38	136.65
RV32IMC_zfinx	1	54	130.79
RV32IMC_zfinx + Perf	7	64	123.3
RV32IM +Perf +Area	7	62	122.47

Table 4 *CoreMark results for each configuration*

Configuration	CoreMark score	CoreMark iterations/sec	Average CPI
RV32I	0.28	14	3
RV32IMC_zfinx	0.52	26	4
RV32IMC_zfinx+Perf	0.77	38	2
RV32IM+Perf +Area	0.66	33	3

Table 5 *Comparison with other RISC-V cores*

RISC-V Core	LUTs	CoreMark score
NEORV32	2,827	0.77
Microblaze	1376	0.48
PicoRV32	1542	0.4
Taiga	1434	2.53
VexRiscv	1418	1.2

5. Conclusion

This research aimed to analyse and evaluate a RISC-V microcontroller system, specifically the NEORV32 core implemented on an FPGA platform. The assessment focused on hardware utilisation and performance. The performance of the processors was compared using different NEORV32 core configurations and CoreMark benchmark tests. In this work, two NEORV32 implementations are evaluated, which are RV32I and RV32I with M and Zfinx extension (RV32I_MC_zfinx). The effect of dedicated hardware and special operations on the performance of the processors is also evaluated on an FPGA board. The experiment results show that RV32I_MC_zfinx consumes 55% and 65% more LUT and registers resources, respectively, compared to the RV32I. Implementing hardware accelerators to RV32I_MC_zfinx results in a 48% increase in CoreMark score. Compared with other existing RISC-V cores, NEORV32 is a good option for embedded system development since it balances performance and resource efficiency for low-power applications.

Acknowledgement

Execution of this research is made possible through the support from various staffs from Faculty of Faculty of Electronic Engineering & Technology, Universiti Malaysia Perlis and Centre of Excellence for Micro System Technology (MiCTEC), Universiti Malaysia Perlis.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Chung Wei Peng and Asral Bahari Jambek;* ***data collection:*** *Chung Wei Peng and Asral Bahari Jambek;* ***analysis and interpretation of results:*** *Chung Wei Peng, Asral Bahari Jambek, Sreedharan Baskara Dass, Lim Teik Wah, Lalin L. Laudis, Avinash Yadlapati, Gnaneshwara Chary Udari, B. Rama Sanjeeva Reddy;* ***draft manuscript preparation:*** *Chung Wei Peng and Asral Bahari Jambek. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] What is Instruction Set Architecture (ISA)? – Arm®. https://www.arm.com/glossary/isa (accessed Jul. 15, 2023).
[2] The Challenges of Making Open-Source RISC-V Deployment Effective - RISC-V International. https://riscv.org/news/2020/04/the-challenges-of-making-open-source-risc-v-deployment-effective/ (accessed Jan. 01, 2023).
[3] Poli , L., Saha , S., Zhai , X., McDonald-Maier , K. D., (2021). Design and Implementation of a RISC V Processor on FPGA. Proceedings - 2021 17th International Conference on Mobility, Sensing, and Networking (MSN), pp. 161–166. https://doi.org/10.1109/MSN53354.2021.00037.
[4] Le , A.T., Dao , B.A., Suzaki, K., and Pham , C. K., (2020). Experiment on Replication of Side Channel Attack via Cache of RISC-V Berkeley Out-of-Order Machine (BOOM) Implemented on FPGA. Fourth Workshop on Computer Architecture Research with RISC-V.
[5] Holler, R., Haselberger, D., Ballek, D., Rossler, P., Krapfenbauer, M., Linauer , M., (2019). Open-Source RISC-V Processor IP Cores for FPGAs - Overview and Evaluation. The 8th Mediterranean Conference on Embedded Computing (MECO). https://doi.org/10.1109/MECO.2019.8760205.
[6] Dörflinger, A. et al. (2021). A Comparative Survey of Open-Source Application-Class RISC-V Processor Implementations. The 18th ACM International Conference on Computing Frontiers. pp: 12–20. https://doi.org/10.1145/3457388.3458657
[7] Heinz, C., Lavan, Y., Hofmann, J., Koch, A. (2019). A Catalog and In-Hardware Evaluation of Open-Source Drop-In Compatible RISC-V Softcore Processors. International Conference on Reconfigurable Computing and FPGAs, (ReConFig). https://doi.org/10.1109/ReConFig48160.2019.8994796.
[8] Bora, S., Paily, R. (2021). A High-Performance Core Microarchitecture Based on RISC-V ISA for Low Power Applications. IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 6, pp. 2132–2136. https://doi.org/10.1109/TCSII.2020.3043204.
[9] Mezger, B. W., Santos, D. A., Dilillo, L., Zeferino, C. A., Melo, D. R. (2022). A Survey of the RISC-V Architecture Software Support. IEEE Access, vol. 10, pp. 51394–51411. https://doi: 10.1109/ACCESS.2022.3174125.
[10] Zheng, T., Cai, G., Huang, Z. (2022). A Soft RISC-V Processor IP with High-performance and Low-resource consumption for FPGA. Proceedings - IEEE International Symposium on Circuits and Systems, pp. 2538–2541. https://doi.org/10.1109/ISCAS48785.2022.9937742.
[11] NEORV32, https://stnolting.github.io/neorv32