Performance Evaluation of PicoRV32 RISC-V Softcore for Resource-Constrained Devices

Marek Jahnke, Lucas Bublitz and Ulf Kulau
Hamburg University of Technology, Germany
marek.jahnke@tuhh.de | lucas.bublitz@tuhh.de | ulf.kulau@tuhh.de

2023 IEEE Nordic Circuits and Systems Conference (NorCAS) | 979-8-3503-3757-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/NorCAS58970.2023.10305479

***Abstract*—In this paper, an extensive analysis of the resource-efficient PicoRV32 softcore, which implements the RISC-V instruction set, is performed. The focus of the analysis is on performance, energy efficiency and resource utilization. For this, both CoreMark and Embench are implemented as benchmark tools.**

In contrast to previous evaluations, a resource-constrained target platform (Low Power (LP) FPGA) is explicitly used for the evaluation, to demonstrate the applicability on very small embedded systems. Furthermore, all configurations of the softcore are investigated, which helps in the estimation and subsequent configuration of the softcore and reveals significant bottlenecks like memory interface.

***Index Terms*—RISC-V, LP FPGA, Softcore, Benchmark, Resource-Constrained Devices**

I. Introduction

Classically, micro controller units (MCUs) are used for tiny embedded systems, e.g. Internet of Things (IoT) devices, smart sensors, or similar. They are very energy efficient, inexpensive, easy to program and highly available. However, for custom requirements, or also the acceleration of computing processes (parallel processing), they offer less flexible hardware structures. For this purpose, the focus of current research is on the growing trend of smaller, more efficient Low Power (LP) Field Programmable Gate Arrays (FPGAs). The class of LP FPGAs, unlike the elements used for high-performance computing, offers small size, high energy efficiency and are also available at low cost.

Classical processing elements, dedicated interfaces and hardware accelerators (e.g., digital filters, FFT, neural networks, coding methods, compression methods) become available in an efficient, parallel structure, but with the cost of less flexibility than a software solution.

Nevertheless, for the orchestration of the signal processing chain, further signal processing or the implementation of flexible protocols, the FPGA offers the advantage of implementing a Central Processing Unit (CPU) as a softcore for the execution of software.

The PicoRV32 [1] provides a softcore compatible with the RISC-V instruction set, which is specifically designed to address the concerns of resource-constrained systems, making it ideal for use in small embedded devices.

In this paper we will investigate how efficient the PicoRV32 can be used as a softcore solution for resource-contrained devices on LP FPGAs. For this purpose, the two benchmarks CoreMark [2] and Embench [3] will be ported to the PicoRV32 and evaluated for different configurations of the softcore. As LP FPGA, the Lattice ICE40 [4] suitable for small embedded devices is used [5]. The choice of the softcore but also the LP FPGA was made deliberately because both the RISC-V and the FPGA toolchain are available as open source [6], [7], making them ideal for academic research, teaching and development. The results of this work help to assess with which configurations the PicoRV32 can be used on a resource-constrained system such as the Lattice ICE40 to be able to use it in small embedded systems.

II. Related Work

The use of FPGAs in small embedded systems such as Smart Sensors or IoT devices is currently trending [8]–[10]. The advantage of FPGAs over classical processing units such as MCUs is the implementation of highly efficient data processing chains, but also special solutions [11], [12]. However, the focus of this paper is on softcores, i.e. CPUs which are also implemented in the FPGA to enable the execution of regular software. Especially the combination of softcore and dedicated hardware accelerators is of immense advantage for the use in Smart Sensors or other embedded systems [13]–[15]. The focus is particularly on resource-efficient softcores, of which a large number already exists [16], [17]. One of these softcores is the PicoRV32, which was specially developed for low resource utilization and implements the RISC-V Instruction Set Architectures (ISA) RV32IMC [1], [18]. In particular, the compatibility with the RISC-V ISA makes this softcore interesting, because there are no licensing problems when using and adapting this softcore for own designs [19], which is particularly advantageous for product-related developments. The efficiency and performance of the PicoRV32 have therefore already been evaluated in other studies. However, mostly this softcore has been evaluated in comparison to other [20], [21]. When looking at these evaluations, it is noticeable that no in-depth investigation of the various configurations of the softcore (ISA extensions) has taken place. Only a single configuration was used without any significant justification for the investigations. Another drawback of existing evaluations is that the target platform was mostly performed away from the FPGAs usually used on embedded systems with high limitation on size, power and costs. As mentioned in Section I, resource-constrained FPGAs are predominantly used here. To

TABLE I
SPECIFICATION OF LP FPGA USED FOR EVALUATIONS [4].

iCE40UP5K	
Logic Cells (LUT + Flip-Flop)	**5280**
EBR Memory Blocks	30
EBR Memory Bits (Kbits)	120
SPRAM Memory Blocks	4
SPRAM Memory Bits (Kbits)	1024
NVCM	Yes
PLL	1
DSP Blocks (MULT16 with 32-bit Accumulator	8
Packages, ball pitch, dimension	**Total User I/O Count**
30-ball WLCSP, 0.4 mm, 2.11 mm × 2.54 mm	21
48-ball QFN, 0.5 mm, 7.0 mm × 7.0 mm	39

fairly evaluate the suitability of the PicoRV32 in the scope of rather small embedded devices, the target platform must also be from the domain of LP FPGAs with limited resources. Both gaps of existing evaluations: i) not considering the different configuration of the ISA extensions and ii) the resource limited target platform in form of a LP FPGA are closed in this paper.

III. LP FPGA EVALUATION PLATTFORM

The evaluation is focused on an LP FPGA with limited resources, which provides a much more realistic assessment of the softcore performance on resource constrained devices.

A. iCE 40 Ultra Plus FPGA (ICE40UP5K)

In particular, a LP *iCE40UP5K* FPGA from the company Lattice is used [4].

The iCE40 UltraPlus FPGA family is characterized, compared to other FPGAs, by a small number of logic blocks. For example, as can be seen from Table I, the iCE40UP5K contains 5280 Logic Cells (LCs) (each consisting of a Look-Up Table (LUT) and a Flip-Flop), where 8 LCs form a logic block and a LUT has K=4 inputs. The iCE40 series requires a core supply voltage of $V_{core} = 1.2V$ and deliberately targets the low-power range. Available as BGA, the size is also remarkable small for tiny embedded devices (cf. Table I). In sum, the iCE40 is suitable as an alternative to small MCUs and thus a well suited candidate for showcase evaluation.

B. iCEbreaker-Board (v1.0e)

The evaluation platform used is the iCEbreaker board (v1.0e), which contains the ICE40UP5K-FPGA, flash memory, and an FTDI for UART communication as well as programming interface [22]. A block diagram is shown in Figure 1. The board comes without extensive peripherals, only LEDs and buttons are present, the PMODS are free for use. Current measurement of I_{core} takes place at V_{core}, so there is no influence of external components (e.g. LED). As non-volatile configuration memory, a 16 MB (about 15.25 MiB) flash memory is connected via SPI. The flash not only contains the FPGA bitstream, but can also be used by the FPGA itself via (Q)SPI interface. The latter is important because the flash is used as instruction memory for the softcore due to the high resource limitations.

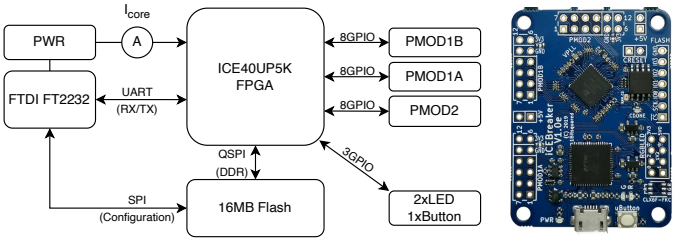

Fig. 1. Icebreaker block diagramm and picture of the used hardware.

IV. PICORV32 - SOFTCORE UNDER TEST

PicoRV32 is a softcore that implements the RISC-V ISA RV32IMC. This can be configured as RV32E, RV32I, RV32IC, RV32IM, or RV32IMC core [1]. Internally, the PicoRV32 has 32KB Random Access Memory (RAM) and has parameters to enable and disable processor features. According to the RISC-V nomenclature (RV<*bit-width*><*extensions*>), this softcore implements 32bit architecture and Base Integer (I), Half the number of integer registers (E), Integer multiplication and division extension (M) and Compressed instruction extension (C) according to the selected configuration. As the extensions might have a significant impact on the overall performance, this paper will evaluate the different configurations extensively.

A. PicoSOC

The existing example implementation PicoSOC is used which implements the PicoRV32 as softcore [23]. The PicoSOC extends the PicoRV32 with further peripheral components, e.g. for Universal Asynchronous Receiver Transmitter (UART) communication, and connects it to the 32 bit system bus. The UART, which is available over the FT2232 USB bridge at host PC, is required for logging of debug-messages as well as the output of evaluation results. Software-wise every peripheral, e.g. UART, Flash-(Q)SPI, as well as GPIO control, is accessible by memory-mapped IO. Due to resource limitations, the flash memory is used for FPGA configuration as well as instruction memory. For the generation of a bitstream and transfer to the flash memory, the open-source synthesis suite *Yosys - Yosys Open SYnthesis Suite* [6] is used. The software is created using the RISC-V compiler [7], which is also open-source.

V. BENCHMARKS

To enable a fair comparison between different platforms and configurations, standardized benchmarks should be used. According to [24], a benchmark should be designed in such a way that it implements typical algorithms and functions in relation to real applications in order to obtain a meaningful result. In this research work, the benchmarks EEMBC CoreMark [2] and Embench [3] are used and described in more detail below.

A. CoreMark

The Embedded Microprocessor Benchmark Consortium (EEMBC) develops benchmarks for various embedded (single-

and multicore) microprocessors [16]. One benchmark of the EEMBC is *CoreMark* [25]. It is based on list processing, such as searching, sorting and flipping and the calculation of CRC to check the data of the list. Furthermore, matrix calculations are performed and a state machine is executed. CoreMark was adapted to run on the PicoRV32 and the output of the benchmark was implemented on the UART. The final result provides the number of benchmark iterations per second. CoreMark recommends to have multiple repetitions of the benchmark, where the number of repetitions r was set to $r = 25$ for this evaluation. Thus, we can calculate the runtime of the entire benchmark by:

$$t = \frac{r}{CoreMark\_Value}[s] \tag{1}$$

B. Embench

Even though CoreMark already covers common functions, it is still considered a *synthetic* benchmark that does not consist of real use cases. Therefore, in 2019 *Embench* benchmark suite was released [16]. The group behind it is involved in the RISC-V Foundation and has as main goal to replace the widely used standard benchmarks Dhrystone and CoreMark with a better solution. For details about implemented bechmark function in Embench, please refer to [3].

The required number of cycles is measured, and the Embench result is given as the required time in milliseconds for an execution of the benchmark at 1MHz. An overall result is calculated from the single results of the individual benchmarks using a geometric average. As the clock rate of the LP FPGA and thus of the softcore is set to `CPU_MHZ`=12MHz, the runtime for the entire benchmark is given by:

$$t = \frac{Embench\_Value}{CPU\_MHZ \cdot 1000}[s] \tag{2}$$

For both benchmarks, CoreMark and Embench, the respective runtime (cf. equation 1 and 2) is later used for the calculation of the energy consumption.

VI. Evaluation

As discussed in Section II, it is required to evaluate different configurations of the softcore in order to get a fair comparison and overall conclusion about the suitability and performance of the PicoRV32 to be used as softcore for resource constrained devices. Table II lists the configurations to be tested. The selected `CPU_MHZ` is 12MHz. In the first column an ID is given, which is used to refer to the respective configuration. The ISA column indicates the instruction set architecture used by the RISC-V compiler. This is directly dependent with the last four parameters `COMPRESSED_ISA`, `ENABLE_MUL`, `ENABLE_FAST_MUL` and `ENABLE_DIV`. If the extension C for compressed instructions is used in the ISA, the parameter `COMPRESSED_ISA` must be activated to be able to interpret the instructions. The last three parameters determine whether a multiplier and a divider module are synthesized. As soon as the extension M is present in the ISA, one of the two `ENABLE_(FAST_)MUL` parameters, as well as `ENABLE_DIV` must be activated. Only one of the two `ENABLE_MUL` or `ENABLE_FAST_MUL` can be activated at a time. Conversely, it would not make sense to activate a hardware multiplier or divider if the extension M is not used in the ISA, since the compiler then does not use the corresponding instructions. The same applies to the `COMPRESSED_ISA` parameter. In addition, the parameters `TWO_STAGE_SHIFT`, `TWO_CYCLE_COMPARE`, `TWO_CYCLE_ALU` and `BARREL_SHIFTER` are available.

TABLE II
List of the softcore configurations of the PicoRV32 used in the evaluation

ID	ISA	CPU_MHZ	TWO_STAGE_SHIFT	TWO_CYCLE_COMPARE	TWO_CYCLE_ALU	BARREL_SHIFTER	COMPRESSED_ISA	ENABLE_MUL	ENABLE_FAST_MUL	ENABLE_DIV
1	I	12	1	0	0	0	0	0	0	0
4	IM	12	1	0	0	0	0	1	0	1
7	IM	12	1	0	0	0	0	0	1	1
19	IC	12	1	0	0	0	1	0	0	0
22	IMC	12	1	0	0	0	1	1	0	1
25	IMC	12	1	0	0	0	1	0	1	1
37	I	12	0	0	0	0	0	0	0	0
40	IM	12	0	0	0	0	0	1	0	1
43	IM	12	0	0	0	0	0	0	1	1
55	IC	12	0	0	0	0	1	0	0	0
58	IMC	12	0	0	0	0	1	1	0	1
61	IMC	12	0	0	0	0	1	0	1	1
73	I	12	0	1	0	0	0	0	0	0
76	IM	12	0	1	0	0	0	1	0	1
79	IM	12	0	1	0	0	0	0	1	1
91	IC	12	0	1	0	0	1	0	0	0
94	IMC	12	0	1	0	0	1	1	0	1
97	IMC	12	0	1	0	0	1	0	1	1
109	I	12	0	0	1	0	0	0	0	0
112	IM	12	0	0	1	0	0	1	0	1
115	IM	12	0	0	1	0	0	0	1	1
127	IC	12	0	0	1	0	1	0	0	0
130	IMC	12	0	0	1	0	1	1	0	1
133	IMC	12	0	0	1	0	1	0	1	1
145	I	12	0	0	0	1	0	0	0	0
148	IM	12	0	0	0	1	0	1	0	1
151	IM	12	0	0	0	1	0	0	1	1
163	IC	12	0	0	0	1	1	0	0	0
166	IMC	12	0	0	0	1	1	1	0	1
169	IMC	12	0	0	0	1	1	0	1	1
181	I	12	1	1	1	1	0	0	0	0
184	IM	12	1	1	1	1	0	1	0	1
187	IM	12	1	1	1	1	0	0	1	1
199	IC	12	1	1	1	1	1	0	0	0
202	IMC	12	1	1	1	1	1	1	0	1
205	IMC	12	1	1	1	1	1	0	1	1

A. Performance Results

After adapting the two benchmarks to the architectures and interfaces of the PicoRV32 and PicoSOC, CoreMark and Embench were executed for all configurations listed in Table II. The results of the evaluation are listed in Figure 2. For better readability, the configurations were divided into blocks according to their ISA extension (I, IC, IM, IMC). In addition, the results are sorted in ascending order of performance, where the ID refers to the respective configuration (cf. Table II).

It can be clearly seen that all configurations with a synthesized hardware multiplier clearly score better in perfor-

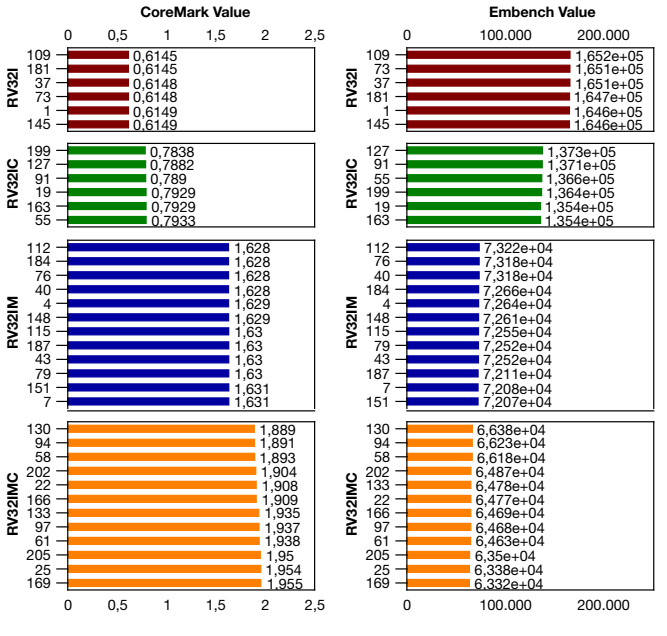

Fig. 2. CoreMark and Embench Results for different configurations of the PicoRV32. Higher CoreMark Value = higher performance. Lower Embench Value = higher performance

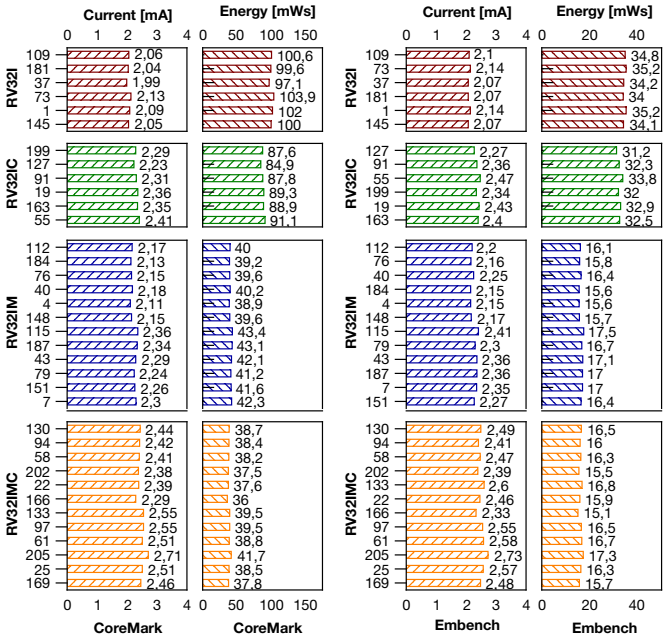

Fig. 3. Average energy consumption when running CoreMark and Embench with different configurations of the PicoRV32. The energy consumption is related to the execution time of the benchmark.

mance than those without. Especially those with the parameter ENABLE_FAST_MUL enabled perform better than the configurations with ENABLE_MUL. A likewise significant performance increase can be seen with COMPRESSED_ISA (I $\rightarrow$ IC 30%, IM $\rightarrow$ IMC 20%). Accordingly, in the first area all configurations with ISA IMC can be found. For the most part, this is also true for the Embench values, where there is a deviation of configuration ID 133, however, which is only marginally slower with ENABLE_FAST_MUL enabled than both configurations ID 22 and ID 166 with only ENABLE_MUL.

The ISA IM without C thus ranks second in performance - again with the advantage of the ENABLE_FAST_MUL over the ENABLE_MUL.

It can be seen that specialized hardware, as well as the compression of the commands is advantageous for the performance. The reason is the instruction memory on the external flash and the resulting bottleneck when reading via the QSPI, which delays the execution. Due to the compressed instruction set, several instructions can sometimes be loaded and executed in one load process, which would otherwise have required two load cycles.

The fact that the extension *M* brings a performance boost is likely due to two reasons: First, the multipliers calculate faster than the loops created by the compiler are executed. Second, since the instruction set used means that significantly fewer instructions are needed and thus less has to be loaded from the flash.

All configurations with the ISA IC are in the second last range and the configurations with the pure base ISA I in the last.

The other parameters have a rather small influence on the performance. Here, the BARREL_SHIFTER performs best, followed by the TWO_STAGE_SHIFT. The parameters TWO_CYCLE_ALU and TWO_CYCLE_COMPARE are rather of disadvantage for the performance, compared to no enabled parameters.

B. Energy Consumption

For evaluation of the energy consumption, the average current I_{core} was measured in parallel to the execution of the benchmarks.

For the CoreMark test runs, the current consumption is between $1.99mA$ and $2.71mA$. The values in the Embench test runs are roughly between $2.07mA$ and $2.73mA$. The highest current consumption is found in all configurations with the ISA *IMC* and ENABLE_FAST_MUL.

However, the energy consumption depends more on the duration of the execution where ENABLE_FAST_MUL performs best. Together with the execution time of CoreMark (cf. Equation 1) and Embench (cf. Equation 2) the respective energy consumption can be calculated as follows (with $V_{core} = 1.2V$):

$$E = V_{core} \cdot I_{core} \cdot t \tag{3}$$

Thus, a strong correlation between better performance (lower execution time t) and the resulting energy can be seen. Figure 3 summarizes all results.

1) Idle and Clock Gating (Sleep): Considering the overall energy consumption, it is important to not just look at active state but also the inactive phase of the softcore. Especially for our targeted small embedded devices, which are e.g. battery powered, the energy consumption during inactive phases is

TABLE III
COMPARISON OF THE PICORV32 SOFTCORE RUNNING ON LP FPGA AGAINST COMMON MCUS USED IN SMALL EMBEDDED DEVICES.

MCU	PicoSOC / picoRV32 ID 37	PicoSOC / picoRV32 ID 169	AVR Atmega2560 [26], [27]	STM32L0 [27], [28]	STM32L4 [27], [29]	esp32-s2 [27], [30]
Familiy	RISC-V	RISC-V	Atmega	Cortex-M0	Cortex-M4	Xtensa
Architecture	32Bit	32Bit	8Bit	32Bit	32Bit	32Bit
Core	picoRV32	picoRV32	ATmega2560	ARMv6-M	Armv7E-M	Xtensa LX7
MHz	12	12	8	32	80	240
Coremark	0,61	1,96	4,25	75,18	265,61	472,81
Coremark/MHz	0,0512	0,1629	0,5313	2,3494	3,3201	1,9700
Sleep Current [μA]	253	256	5	0,27	0,12	22
Active Current [mA]	1,9900	2,4600	7,2000	6,3000	10,2000	23,0000
Current/MHz	0,1658	0,2050	0,9000	0,1969	0,1275	0,0958
CoreMark/mA	0,3089	0,7947	0,5903	11,9333	26,0402	20,5570

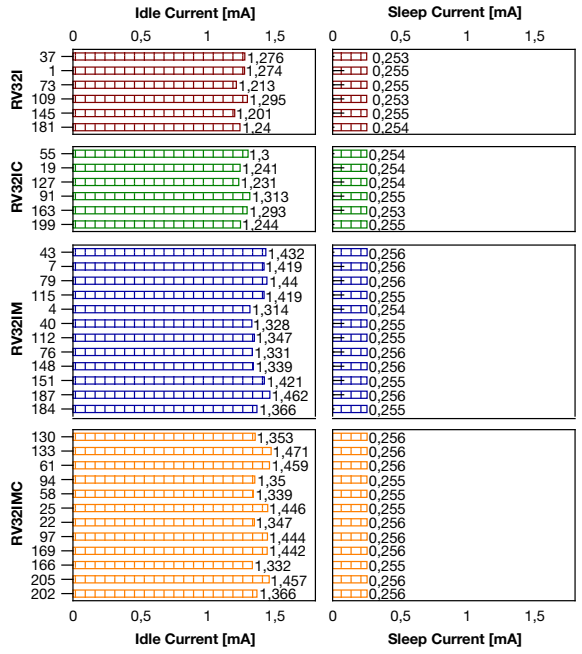

Fig. 4. Average current consumption in idle state as well as in power gated state (sleep) with different configurations of the PicoRV32.

of high importance too. In contrast to MCUs, the FPGA per se does not offer any sleep states. For this reason we investigated the current consumption during idle state, which means processor clock is active but idle waiting. In addition, clock gating can be applied where the clock is removed from the PicoSOC putting the softcore into quasi-sleep-state.

Figure 4 summarizes the power consumption in idle state and also in sleep state for the different configurations. For the sleep state (clock gating) a current of $0.255 \pm 8.5 \cdot 10^{-7} mA$ was measured independent of the selected configuration.

For the power consumption in idle state, a slight dependency on the selected ISA extension can be seen, but again no clear connection to the softcore size (resources). However, the current is between 1.201mA and 1.471mA, a difference of 22%. Thus, depending on the actual duty cycle, the careful selection of a specific configuration can improve the overall energy consumption.

VII. COMPARISON WITH COMMON MCUS

In terms of performance and efficiency, a softcore on an FPGA cannot match an MCU. Nevertheless, to obtain a classification in relation to common MCUs, the results of this study are compared with MCUs, which are usually used for small embedded systems (AVR Atmega2560 [26], STM32L0 [28], STM32L4 [29] and esp32-s2 [30]). Since there is a high correlation of results between CoreMark and Embench, this study is limited to the CoreMark benchmark. The relevant results of the benchmarks are available here [27]. For the other parameters of the MCUs, the respective data sheets were used. The configurations ID 37 (lowest power consumption) and ID 169 (highest performance) were selected for the PicoRV32.

Table III summarizes the results of the comparison. Unsurprisingly, the softcore on the LP FPGA is inferior to all MCUs in terms of performance. Only the efficiency (CoreMark/mA) is slightly better than the Atmega2560 with the configuration ID 169.

A. FPGA Resources

Figure 5 shows the number of LUTs. Once again, the difference between the individual ISA configurations can be seen, but not as distinguished as for the performance analysis and the energy consumption. ID 32 requires the least with 3252 LUTs and ID 202 with 5148 LUTs the most resources. The used FPGA has a maximum of 5280 LUTs available (cf. Table I), so that this LP FPGA used for ID 202 is utilized to 97.6% so that no further HW resources are available for e.g. accelerators or interfaces.

Looking at the total number of LUTs used, it can be seen that again the combination of extensions M and C leads to a higher resource utilization. The use of extension M leads to a significant increase in the number of LUTs, as can be seen by the jump from the transition from configuration ID 199 without M to configuration ID 43 with M. However, the extension M in turn is also responsible for the performance gain (cf. Section VI-A.

VIII. CONCLUSION

This paper was motivated by the increasing use of LP FPGAs on small embedded systems sensors, IoT-Devices, etc.

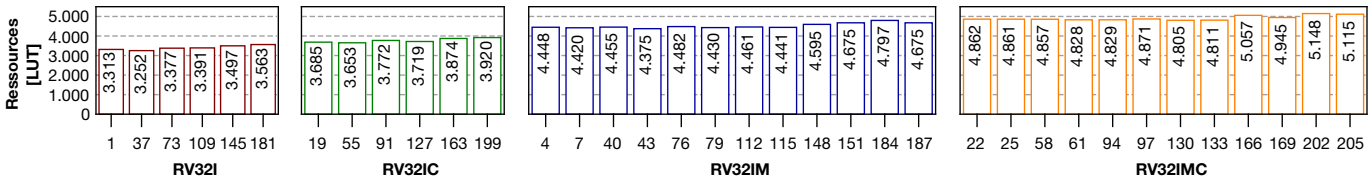

Fig. 5. Required resources (LUTs) for the respective configuration of the PicoRV32 on the ICE40UP5K LP FPGA

In order to be able to run software in addition to the advantages of configurable hardware, resource-efficient softcores are often used. One of these lightweight softcores is the PicoRV32, which implements the RISC-V instruction set and has been extensively evaluated for this work.

To get a holistic picture of the evaluation and in contrast to previous evaluations, an LP FPGA (iCE40UP5k) was chosen as the target platform and all configurations of the instruction set were evaluated. In addition, two recognized benchmarks (CoreMark and Embench) were run.

Thus, the actual performance of the softcore on a resource-constrained processing unit was evaluated, showing that extensions such as multipliers M increase the performance and a compressed instruction set C mitigates the memory bottleneck in external program memories. The power consumption was also examined, whereby the active power consumption, as well as that in idle state, and in sleep state (clock gating) legitimize the softcore for use in low power applications.

Nevertheless, the comparison with commonly used MCUs shows that both performance and power efficiency of the PicoRV32 on the LP FPGA are significantly lower and the FPGA cannot show its advantage of configurable hardware by using a soft core alone, but rather dedicated accelerators and custom hardware make up the immense advantage for resource-constrained systems.

ACKNOWLEDGEMENT

This work has received funding from the German Aerospace Center (DLR) in the project ‘SatelLight’ under the grant number 50RP2360B.

REFERENCES

[1] YosysHQ, “PicoRV32 - A Size-Optimized RISC-V CPU.” https://github.com/YosysHQ/picorv32.
[2] “Embedded microprocessor benchmark consortium (eembc), CoreMark benchmark.” https://www.eembc.org/coremark/. Accessed: 2023-02-27.
[3] “Embench benchmark.” https://www.embench.org/. Accessed: 2023-02-27.
[4] Lattice Semiconductor, “iCE40 UltraPlus Family Data Sheet, Rev. 2.0.” https://tinyurl.com/52ve98h6. Accessed: 2023-08-20.
[5] A. De La Piedra, A. Braeken, and A. Touhafi, “Sensor systems based on fpgas and their applications: A survey,” *Sensors*, vol. 12, no. 9, pp. 12235–12264, 2012.
[6] “Yosys – Yosys Open SYnthesis Suite.” https://github.com/YosysHQ/yosys. Accessed: 2023-02-27.
[7] “riscv-tools.” https://github.com/riscv-software-src/riscv-tools. Accessed: 2023-02-27.
[8] S. M. Aziz, D. H. Hoskin, D. M. Pham, and J. Kamruzzaman, “Remote reconfiguration of fpga-based wireless sensor nodes for flexible internet of things,” *Computers and Electrical Engineering*, vol. 100, p. 107935, 2022.
[9] W.-P. Kiat, K.-M. Mok, W.-K. Lee, H.-G. Goh, and R. Achar, “An energy efficient fpga partial reconfiguration based micro-architectural technique for iot applications,” *Microprocessors and Microsystems*, vol. 73, p. 102966, 2020.
[10] D. M. Pham and S. M. Aziz, “Flexis—a flexible sensor node platform for the internet of things,” *Sensors*, vol. 21, no. 15, p. 5154, 2021.
[11] B.-L. Tan, K.-M. Mok, J.-J. Chang, W.-K. Lee, and S. O. Hwang, “Risc32-lp: Low-power fpga-based iot sensor nodes with energy reduction program analyzer,” *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4214–4228, 2021.
[12] U. Kulau, J. Rust, D. Szafranski, M. Drobczyk, and U.-V. Albrecht, “Differential bcg sensor system for long term health monitoring experiment on the iss,” in *2022 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, IEEE, 2022.
[13] B. Bengherbia, M. O. Zmirli, A. Toubal, and A. Guessoum, “Fpga-based wireless sensor nodes for vibration monitoring system and fault diagnosis,” *Measurement*, vol. 101, pp. 81–92, 2017.
[14] P. Biswas, S. Banerjee, N. Dutt, P. Ienne, and L. Pozzi, “Performance and energy benefits of instruction set extensions in an fpga soft core,” in *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID’06)*, pp. 6–pp, IEEE, 2006.
[15] D. W. Todd, *Tightly coupling the PicoRV32 RISC-V processor with custom logic accelerators via a generic interface*. PhD thesis, Clemson University, 2021.
[16] C. Heinz, Y. Lavan, J. Hofmann, and A. Koch, “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)*, pp. 1–8, IEEE, 2019.
[17] P. Kalkundri, H. Guhilot, and K. Ravi, “Soft embedded cores for fpga,” in *ICT Systems and Sustainability: Proceedings of ICT4SD 2021, Volume 1*, pp. 895–905, Springer, 2022.
[18] I. Elsadek and E. Y. Tawfik, “Risc-v resource-constrained cores: A survey and energy comparison,” in *2021 19th IEEE International New Circuits and Systems Conference (NEWCAS)*, pp. 1–5, IEEE, 2021.
[19] M. Sharma, E. Bhatnagar, K. Puri, A. Mitra, and J. Agarwal, “A survey of risc-v cpu for iot applications,” *Available at SSRN 4033491*, 2022.
[20] R. Höller, D. Haselberger, D. Ballek, P. Rössler, M. Krapfenbauer, and M. Linauer, “Open-source risc-v processor ip cores for fpgas—overview and evaluation,” in *2019 8th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–6, IEEE, 2019.
[21] T. Zheng, G. Cai, and Z. Huang, “A soft risc-v processor ip with high-performance and low-resource consumption for fpga,” in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2538–2541, IEEE, 2022.
[22] “iCEBreaker FPGA.” https://github.com/icebreaker-fpga/icebreaker.
[23] “YosysHQ, PicoSoC - A simple example SoC using PicoRV32.” https://github.com/YosysHQ/picorv32/tree/master/picosoc. Accessed: 2023-02-27.
[24] I. Tsekoura, G. Rebel, P. Glosekotter, and M. Berekovic, “An evaluation of energy efficient microcontrollers,” in *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, IEEE, may 2014.
[25] S. Gal-On and M. Levy, “Exploring CoreMark™ – A Benchmark Maximizing Simplicity and Efficacy.” The Embed-ded Microprocessor Benchmark Consortium.
[26] Microchip, *Datasheet: Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V*, 2014. Rev. 2549Q–AVR–02/2014.
[27] CoreMark, “Scores.” https://www.eembc.org/coremark/scores.php. Accessed: 20.08.2023.
[28] S. Microelectronics, *Datasheet: STM32L051x6 STM32L051x8*, 2019. Rev. DS10184 Rev 10.
[29] S. Microelectronics, *Datasheet: STM32L476xx*, 2019. Rev. DS10198 Rev 8.
[30] espressif, *Datasheet: SoC with Xtensa SingleCore 32bit LX7 Microprocessor*, 2023. Rev. Version 1.5.