The Scale4Edge RISC-V Ecosystem

Wolfgang Ecker
Infineon Technologies AG
wolfgang.ecker@infineon.com

Milos Krstic
IHP –Leibniz Institut für innovative Mikroelektronik & University Potsdam
krstic@ihp-microelectronics.de

Andreas Mauderer
Robert Bosch GmbH
andreas.mauderer@de.bosch.com

Eyck Jentzsch
MINRES Technologies GmbH
eyck@minres.com

Mihaela Damian, Julian Oppermann, Andreas Koch
Technical University of Darmstadt
firstname.lastname@esa.informatik.tu-darmstadt.de

Peer Adelt, Wolfgang Mueller
Paderborn University
firstname.lastname@hni.upb.de

Vladimir Herdt, Rolf Drechsler
University of Bremen / DFKI GmbH
firstname.lastname@uni-bremen.de

Rafael Stahl, Karsten Emrich, Daniel Mueller-Gritschneder
Technical University of Munich
firstname.lastname@tum.de

Jan Schlamelcher, Kim Grüttner
OFFIS - Institut für Informatik
firstname.lastname@offis.de

Jörg Bormann
Siemens EDA
joerg.bormann@siemens.com

Wolfgang Kunz
Technische Universität Kaiserslautern
wolfgang.kunz@eit.uni-kl.de

Reinhold Heckmann
AbsInt Angewandte Informatik GmbH
heckmann@absint.com

Gerhard Angst, Ralf Wimmer
Concept Engineering GmbH
firstname.lastname@concept.de

Bernd Becker, Philipp Scholl
Albert-Ludwigs-Universität Freiburg
firstname.lastname@informatik.uni-freiburg.de

Paul Palomero Bernardo, Oliver Bringmann
Universität Tübingen
firstname.lastname@uni-tuebingen.de

Johannes Partzsch, Christian Mayr
Technische Universität Dresden
firstname.lastname@tu-dresden.de

***Abstract*—This paper introduces the project Scale4Edge. The project is focused on enabling an effective RISC-V ecosystem for optimization of edge applications. We describe the basic components of this ecosystem and introduce the envisioned demonstrators, which will be used in their evaluation.**

Keywords—RISC-V, edge applications, ecosystem

I. Introduction

Internet of Things (IoT) applications already became the reality around us over the last several years. In many applications, including automotive, industry automation, and space, IoT solutions are needed and applied. IoT devices rely frequently on edge processing, which needs to perform efficiently, with respect to energy but also from the cost point of view. Moreover, for many applications, the aspects of safety, security, and reliability are equally important as performance or energy consumption.

For application-specific processing in IoT devices it is advisable to focus on specifically optimized processing architectures, which could provide up to 10 times better performance for such demanding applications. RISC-V has been introduced some years ago as a novel scalable and extendible Instruction Set Architecture (ISA), which could provide competitive solutions for special edge processors.

Nevertheless, to enable effective use of RISC-V architectures for edge applications, the challenge is to provide a complete tool/IP ecosystem, which enables the implementation, verification, and test of highly scalable application-specific edge components. The **Scale4Edge project** addresses these needs and proposes an integrated RISC-V ecosystem with HW and SW support for safe, secure, and reliable applications.

The main objective of the Scale4Edge project is the development of an ecosystem, based on a platform concept, to supply efficient and cost-effective application specific edge devices and value-added services addressing different market segments. This will be achieved through the automatic and very fine-grained adaptation of highly generic components to the application. The Scale4Edge ecosystem covers highly scalable components and tools and extends them for application-specific edge components at three levels: (1) CPU instruction level defined by the RISC-V Instruction Set Architecture (ISA), (2) software level defined by the latest C programming language standard C11 with compilers and libraries open to complementary standards like MISRA-C, and (3) operating system and firmware level through system services, configuration interfaces, and drivers.

The ecosystem platform is developed as customizable to the individual application, through the RISC-V ISA, which may define optional custom instructions, e.g., to support non-interruptible instructions, for individual applications. As such, Scale4Edge is based on a broadly scalable hardware addressing different pipeline architectures, multi-core architectures, co-processors and hardware accelerators, like for AI and DSP applications. In terms of hardware scalability, additions for non-functional properties such as energy efficiency, fault tolerance, robustness, and safety and security are important contributions of the Scale4Edge ecosystem. Various measures such as hardening of registers by error correction bits, scaling of clock frequency or protection of memory areas by an MPU are examples. Targeted main applications of Scale4Edge are classical edge components at the interface of the cloud or fog, respectively. Moreover, more specific components, such as sensor interfaces for automobiles and high-reliable (HighRel) electronics for reliability critical applications are considered as well.

The remainder of this paper continues with an introduction to the Scale4Edge ecosystem, its components, and their interaction before applications and demonstrators are presented and a final summary and conclusion is given.

II. The Ecosystem

The Scale4Edge ecosystem is composed of a large set of interacting tools to customize, design, verify, and produce application specific RISC-V based microprocessors. Figure 1 gives an overview of the different components, their interaction and how they are aligned to the different phases of the design and manufacturing process, which are introduced in more details in the following sections.

The ISA description and specification language CoreDSL is the key to define custom ISA extensions and variants of the

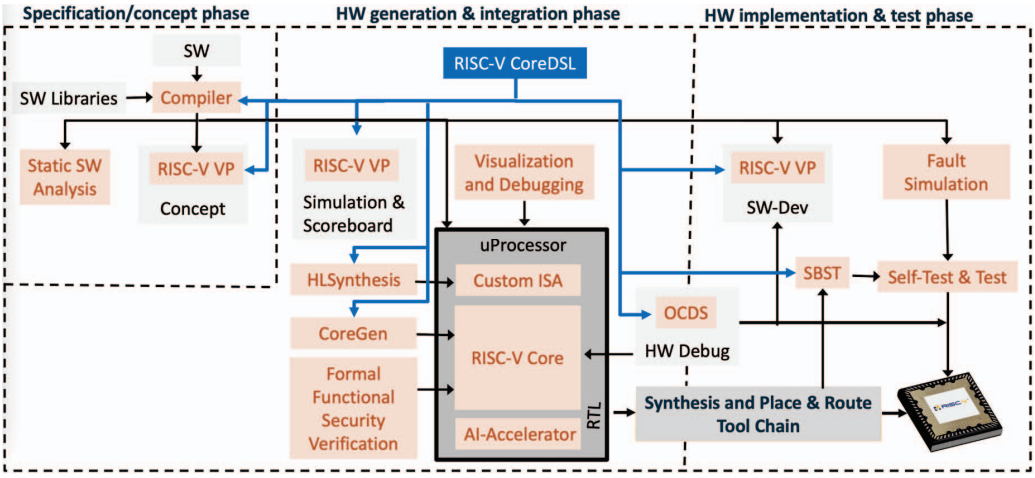

Figure 1: Scale4Edge ecosystem overview.

RISC-V core, which are both generated. Other analysis and verification tools are customizable by the CoreDSL. Tools of the ecosystem cover relevant parts of the complete Hardware/Software design process over hardware/software verification and debugging until final tapeout and chip test. For early software analysis and validation, the ecosystem is based on virtual prototypes, which are applied for ISA specification, simulation, and software development. This smoothly complements and extends the industrial synthesis and Place & Route design flow by commercial EDA tools.

A. *ISA/Core Specification: CoreDSL*

To ensure a consistent and comprehensive methodology in RISC-V ISA development, it is important to use a unified description of the supported instructions and their behavior. This ensures that all tools, including generators for RTL IP, virtual platforms, formal verification tools or compilers, use the same ISA specification.

For this purpose, an existing but rudimentary description language was further developed. This language, called CoreDSL (Version 2) permits the modular description of the architectural state and the instructions. In order to keep the entrance barrier low, a C syntax was selected for the description of the state and the instructions. In addition, some extensions, like arbitrary sized data types and bit-select operations, are used. To describe concurrent operations, a new keyword 'spawn' has been added. Other existing keywords like 'extern' or 'register' got explicit meaning.

A CoreDSL description consists of at least two sections: the definition of the architectural state and the definition of the instructions. Figure 2 shows a simple example of a processor core with 16-bit registers and a 64 kB memory address space. CoreDSL V2 does not yet support microarchitecture element descriptions. This will be the focus of future work and extensions of the language. Use cases for this would be, e.g., the generation of performance models and the early estimation of the influence of custom instructions on the overall performance of a given software. The complete specification of CoreDSL, as well as a reference implementation of the grammar based on XText is available as Open Source[1]. This allows easy usage for new use cases. In the Scale4Edge project, different transformations were and are implemented, e.g., to synthesize custom instructions for the MINRES pipeline, to generate the corresponding instruction set simulator, as well as compiler or to generate the properties for the formal specification.

simple.core_desc ×

```
1  Core Simple {
2    architectural_state {
3     register unsigned short R[16];
4     extern unsigned char mem[1<<16];
5    }
6    instructions {
7     MOV{
8         encoding: 0b00 :: src[3:0] :: tgt[3:0] :: 0b110011;
9         behavior: {
10            R[tgt] = R[src];
11        }
12     }
13    }
14 }
```

Figure 2: CoreDSL description example.

B. *High-Level Synthesis*

Implementing custom instructions requires expertise in hardware design as well as intricate knowledge of the host core. In the Scale4Edge ecosystem, we aim to make such ISA extensions accessible to non-experts, by providing tools to automate most of the process using high-level synthesis (HLS) techniques. Specifically, we synthesize a custom hardware architecture from the descriptions of instruction behavior in the CoreDSL source file and use the supplied metadata to automatically integrate the generated module into the host core's pipeline.

In order to support instructions of different complexities, ranging from simple combinational operations with two register operands, to decoupled functional units, we are designing a lean, specialized HLS engine. Our primary design goal is the efficient use of chip area, a key requirement in the edge computing setting. To achieve that goal, we employ provably optimal static pipelining combined with resource-sharing whenever possible. The interface between the synthesized extension module and the host core is

[1] https://github.com/Minres/CoreDSL/wiki/CoreDSL-2-programmer's-manual

automatically adapted, and hence supports exactly the functionality actually required by the current set of extensions. Lastly, the HLS algorithms will present multiple trade-off solutions to the designer, to let them choose the best one for their application.

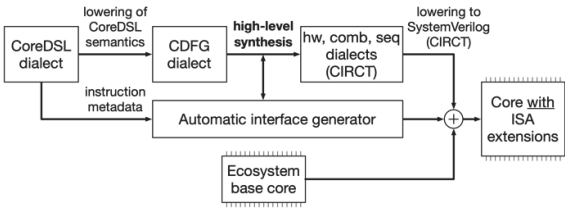

Figure 3: High-level synthesis flow for ISA extensions.

The HLS flow is outlined in Figure 3. The approach is based on CIRCT[2], an LLVM Incubator Project providing a toolkit for implementing hardware design tools built on top of the popular MLIR compiler framework.

C. Software Analysis and Synthesis

1) Static Software Analysis

Static software analysis means analyzing the software without actually executing it. This allows for obtaining useful information about the software that is valid for all executions with all possible inputs. In general, such information cannot be obtained by testing since it is usually impossible to prove that all inputs have been covered. AbsInt offers three static analysis tools for inclusion into the Scale4Edge ecosystem.

Astrée analyzes C source code with the goal to find possible runtime errors or proving their absence. It is therefore largely independent from the target architecture except when very target-specific software should be analyzed. Scale4Edge develops methods for improving the analyzability of such software in Astrée.

StackAnalyzer analyzes binary executables for determining their worst-case stack usage. This information is important to prove the impossibility of stack overflow, which could cause the system to crash or behave erroneously. Since StackAnalyzer operates on binary executables, it has to be adapted to each new target architecture such as RISC-V. This requires implementing a decoder for reading RISC-V executables and a value analysis for determining the possible values of memory cells and registers (including the stack pointer) at every program point. This value information is also used for finding conditions that are always true or always false, and for finding the possible targets of computed calls and branches.

aiT analyzes binary executables for determining their worst-case execution time. This information is important for proving that real-time tasks are completed within their specified time frame. Operating on binary executables, aiT has to be adapted to the RISC-V architecture. Apart from decoder and value analyzer as described above, this involves the implementation of a cache and pipeline analyzer that records all possible cache and pipeline states. These states heavily influence the timing behavior; for instance, cache misses take much more time than cache hits. This part of the analyzer needs exact knowledge about the timing behavior of the RISC-V chip used in the project.

2) Compiler Synthesis and SDK Generation

To exploit the powerful dynamic nature of RISC-V Instruction Set Architecture Extensions (ISAX), equally dynamic tooling support is needed, e.g., by corresponding compiler frameworks. Based on CoreDSL V2, the project will implement a compiler generator to produce this tooling based on such CoreDSL definitions.

Work has already gone into implementing RISC-V support in existing compilers, most importantly in the form of GCC- and Clang/LLVM based RISC-V toolchains. These compilers, or, more specifically, compiler frameworks, implement all modern features of the C/C++ language to produce highly efficient and optimized code, making the de-facto-standard in state-of-the-art software development. This leads to well supported integration in existing applications and platforms. Especially Clang/LLVM can be considered a popular choice for these kinds of tasks due to its extensible nature and its product-oriented software license.

Since the project aims to automate the production of such compiler extensions based on a CoreDSL ISAX definition, the project plans to implement its tooling using the more flexible Clang/LLVM instead. LLVM already implements many parts of the ISA using a custom format called TableGen instead of hand-crafted C/C++ code. The project plans to provide tooling support for translating the relevant aspects of a CoreDSL ISAX definition into the TableGen format and generate the actual C/C++ code for implementing it using the existing Clang/LLVM tools (see Figure 4).

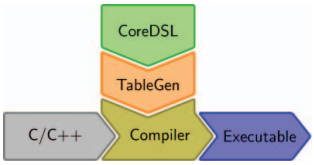

Figure 4: C/C++ Compiler generation based on CoreDSL.

Such a combination of dynamic tooling based on existing and state-of-the-art compiler technology allows us to ease the production of experimental compilers for researching the performance of competing CoreDSL definitions regarding optimization, and to provide sufficiently mature tools for implementing safety critical applications that guarantee the correct translation of at least a subset of the C programming language.

D. Verification and Simulation

1) Virtual Prototype Simulation

VPs (Virtual Prototypes) serve as platforms for concept engineering (Concept VP), early SW development (SW-Dev VP), and functional reference models for the HW design flow (Simulation and Scoreboard VP). To facilitate and speed-up the VP creation process, we are developing the VP-VIBES framework[3] [1] that provides a set of common building blocks for VPs. An important part is the CoreDSL support that enables to generate an ISS based on the CoreDSL description to boost prototyping activities with custom instruction sets. This allows to build advanced VPs for simulation of RISC-V based systems very efficiently [1].

For verification purposes we developed two VP-driven approaches for RISC-V SW and RTL processor verification,

[2] https://circt.llvm.org

[3] https://github.com/VP-Vibes

respectively. The first approach integrates concolic testing with a VP-based simulation environment [2]. Concolic testing essentially works by tracking symbolic execution constraints alongside the SW execution in order to generate new test cases that successively maximize the SW path coverage. The approach integrates seamlessly with the TLM-2.0 bus communication standard and is tailored for the RISC-V ISA. The basic concolic testing approach was extended to detect spatial memory safety violations and found several new bugs in the RIOT OS. In addition, the detection of stack overflows was investigated. The second approach integrates a RISC-V instruction stream generator in a co-simulation setting with an RTL processor under test and an ISS reference model [3]. The tight co-simulation and custom randomized generator enable a very efficient and comprehensive testing process. We found several intricate bugs in the Scale4Edge ecosystem core using the approach.

2) Formal Verification for Functional Correctness and Security

The Scale4Edge ecosystem is an example for the increasing role of customizable and extendible HW and SW infrastructures involving components from numerous providers as well as from open-source domains. This calls for advanced and highly effective verification methods. Scale4Edge meets the wide-spread concerns that subtle details of the RTL implementation, as they result from seemingly innocuous design decisions, can lead to severe security vulnerabilities in HW platforms. Especially, with the trend towards widely distributed design activities in the context of open-source initiatives, not only design bugs but also the risk of "backdoors" or "trojans" must be given appropriate attention [4]. To this end, in addition to conventional verification methods, the Scale4Edge safety and security assurance rests on two new pillars.

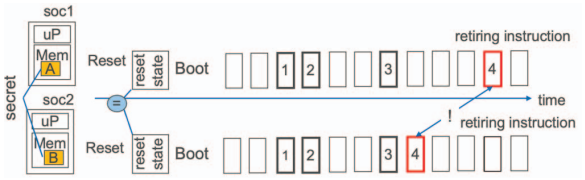

Figure 5: UPEC on unrolled SoC model.

Siemens EDA automates its formal GapFree verification approach for processor cores[4], which guarantees for a given ISA specification the complete coverage of all relevant processor behaviors. Such functional verification of the core is powerful enough to even detect trojans despite their low activation probability. It is complementary with the following second pillar of S4E HW security. The ecosystem integrates "Unique Program Execution Checking (UPEC)", a new formal verification technique was jointly researched. UPEC is the first tool of its kind. Security violations are detected by comparing two instances of the same SoC running any but the same program. Only the secret information may vary between the two instances. UPEC is based on checking certain equivalences and timing conditions between selected state variables on a bounded model (Figure 5). This allows for detecting security-critical design bugs in the entire SoC as well as side channels and backdoors. A commercialization of these contributions to Scale4Edge security assurance is planned. A first prototype of UPEC demonstrates its potential [5].

3) Fault Coverage and Fault Simulation

The Scale4Edge project also applies chip tests by the execution of compiled C programs. To increase their fault coverage, compiled C programs are analyzed and simulated on a QEMU based VP, which has been extended for RISC-V fault coverage analysis and fault simulation [6]. Realistic conclusions with respect to their coverage at RTL can be drawn as we can identify common structures in the VP and the corresponding RTL model like the executed instruction, the GPRs, and the CSRs, and trace them to gate level and layout. As the VP execution is based on principles of Just-in-Time compilation the speed heavily depends on the structure of the executed software. In our applications, the execution speed typically ranges from 0.5 to 50 MIPS x n where n stands for the number of cores of the simulation host. Such execution speeds allow to extend the analysis beyond single stuck-at-fault assumptions to a restricted multi stuck-at fault analysis. Moreover, it also allows a restricted transient fault analysis of fault tolerant architectures as that also multiplies the number of required fault injections and simulation runs, i.e., mutations, under SEU (Single Event Upset) assumptions.

Figure 6 gives an overview of our scalable framework. The ISA module configuration customizes the software generator, the compiler as well as the fault coverage analysis and injection. The framework comes with a library with RISC-V test suites like the RISC-V architectural test framework and the RISC-V unit tests, and scalable generators like Torture and Csmith. After a first analysis, the set of programs is reconfigured to finally arrive at a minimum set of programs with maximum coverage.

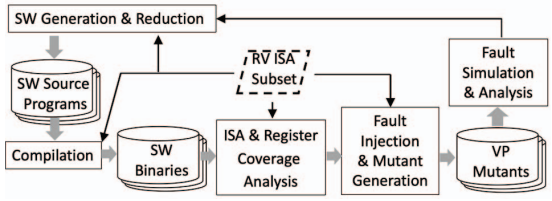

Figure 6: Scalable framework for fault coverage and simulation.

4) SBST Generation

The adaptability of the Scale4Edge Ecosystem to a broad range of constraints w. r. t. efficiency, security, safety, and computational power requires that also the test process can be scaled accordingly. We target a test strategy to find manufacturing defects during production, and moreover, to detect degradation in the field. An essential part will be *Software-Based Self-Tests* (SBST). We developed an approach for the automatic creation of SBST programs for RISC-V architectures via SAT-based test pattern generation. Not only stuck-at-faults, but also cell-aware fault models are supported. The SBST programs can be run after production to detect manufacturing defects and, during idle times, degradation in the field.

[4] www.onespin.com/solutions/risc-v

5) Visualisation and Debugging

The Scale4Edge Ecosystem includes the powerful visualization and debugging engine StarVision PRO[5] that is adapted to the specific needs of the ecosystem. For instance, we extend the tool to visualize not only test data and the propagation of fault effects, but also the test coverage in different hardware modules to assist the user in identifying critical hard-to-test parts of a RISC-V system where testability needs to be improved (see Figure 7). Additionally, we support the interactive selection of a subset of stuck-at and cell-aware faults for which tests should be generated.

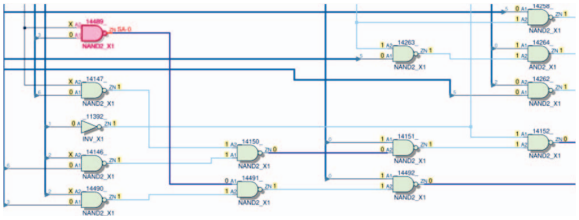

Figure 7: Visualization of a test pattern for a stuck-at-fault in Dark-RISC-V. The affected gate and the propagated fault effect are shown using different colours.

III. Applications and Demonstrators

The Scale4Edge project addresses three areas for demonstration: the ecosystem demonstrators, demonstrators for AI applications, and a HiRel demonstrator.

A. Ecosystem Validation Demonstrator

The Scale4Edge ecosystem is applied to two different automotive applications to validate the ecosystem. The first application are sensor signal processing ASICs, where on-chip processors execute typical DSP algorithms as well as safety monitoring and communication protocol tasks. The second application is a siren detection system, where audio signals are processed by ML algorithms running on an on-chip processor.

For both applications, demonstrators are built by using the Scale4Edge ecosystem. Custom extensions beneficial for each application are identified and specified using the CoreDSL. Based on this description, hardware implementations of the custom extensions are automatically generated and integrated into the ecosystem core. At the same time, the extendible compiler is automatically extended based on the CoreDSL description. The existing application software is then ported to the Scale4Edge ecosystem core. For the software bring-up, Virtual Platforms are built incorporating the DBT-RISE processor model. The siren detection application is also used to build a physical demonstrator. Here, a MEMS microphone is connected to the Scale4Edge ecosystem chip and the siren detection NN software is executed on the integrated ecosystem core.

The ecosystem demonstrator chip is developed in two version. In a first step, an area-optimized version of a generated RISC-V IMC core has been integrated into the PULPissimo [6] platform with 2x32kB SRAM. Synthesis, physical implementation, and signoff has been conducted with industrial tools from Cadence and Siemens EDA for tapeout in May 2021 through Europractice with 22FDX technology from Global Foundries. Figure 8 shows the chip layout of the ecosystem demonstrator Version 1. Version 2 of the ecosystem demonstrator is planned as a fully-fledged chip for siren detection application with AI accelerator in 2022.

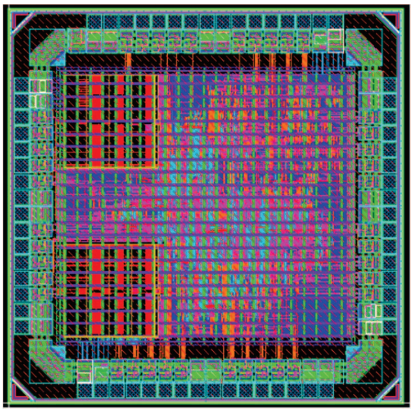

Figure 8: *Ecosystem demonstrator Version 1 layout.*

B. AI Edge Processing Demonstrators

1) TinyML

The ecosystem includes a complete end-to-end flow, i.e., TinyML, that transforms a given machine learning model into deployable machine code with VP-VIBES support. TinyML deployment use and extend two frameworks. Firstly, the *TensorFlow Lite for Microcontrollers (TFLM)* can be used in an automated flow using their interpreter-based approach. A variant of that flow is given with a static code generation approach that eliminates the overheads of the interpreter. Secondly, the *TVM* framework is supported in two variants: a static code generation approach similar to the *TFLM* one, and the *TVM Ahead-of-Time (AOT)* backend. The latter is also a code generation approach, but was only recently developed and not yet matured for productive use. While the *TFLM* framework with *muriscv_nn* support currently outperforms *TVM*, the increased flexibility and control of intermediate steps in TVM allows for more powerful transformations that may lead to a superior performance. This includes plans to utilize global data flow transformations to reduce memory requirements.

2) Parameterizable ML Accelerator

The ecosystem includes the scalable hardware accelerator *UltraTrail* for application-specific ultralow-power edge AI processing [7]. Figure 9 depicts the top-level architecture. The accelerator uses distributed memories to store the features (FMEM0-2), parameters (W/BMEM) and local results (LMEM). Features, parameters, and internal results

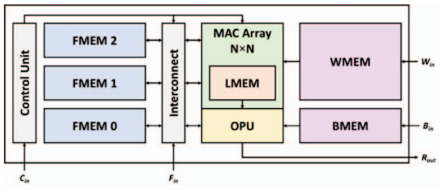

Figure 9: Overview of the UltraTrail architecture.

[5] https://www.concept.de/StarVision.html

[6] https://github.com/pulp-platform/pulpissimo

use a fixed-point representation. An array of multiply and accumulate (MAC) units calculates the convolutional and fully-connect layers. A separate output processing unit (OPU) handles post-processing operations like bias addition, application of a ReLU activation function, or average pooling. The architecture of UltraTrail has been parameterized such that it can be easily fit to various ML workloads. Memory sizes, word widths, the MAC array dimension, and supported post-processing operations can be automatically configured using a hardware/software co-design process. UltraTrail is integrated into the PULPissimo platform as a co-processor and will be part of Version 2 of the ecosystem demonstrator.

3) SpiNNedge Accelerator

The SpiNNedge accelerator module allows to offload signal processing and recurrent neural network (RNN) workloads from the RISC-V processor. Windowing, filtering and frequency transforms (FFT, DCT) are supported, and can be cascaded to more complex preprocessing stages such as Mel-Frequency Cepstral Coefficients (MFCC) for audio processing. Different kinds of RNNs, such as GRU, LSTM or LMU, are supported. Processing effort in the RNN is significantly reduced via delta encoding and subsequent sparsity exploitation. The accelerator follows a loosely-coupled approach, but still shares its main memory with the processor. An internal control module orchestrates the different calculations and enables the accelerator to perform complex processing layers autonomously. A TVM backend for the accelerator is planned for simple ML model deployment and integration with the ML activities in the Scale4Edge ecosystem.

C. HiRel Demonstrator

High Reliability (HiRel) applications are also one important application field of the RISC-V ecosystem. The relevant applications include space, high altitude avionics, nuclear etc. An additional requirement which needs to be fulfilled for such applications is related to resilience to soft errors, i.e., Single Event Effects (SEEs). This requirement is usually addressed at different abstraction layers, starting from technology and standard cell level, providing certain protection against SEEs. Nevertheless, additional efforts are required at the higher abstraction layers in order to enable optimal trade-off between performance, energy consumption and reliability. In order to enable dynamicity and reconfiguration of redundant resources, the Scale4Edge HiRel platform employs a multi-processor architecture of RISC-V cores, that could be dynamically configured in hardware lock-step modes, enabling core level error detection and correction. Additionally, further operating modes are available, including high performance mode, supporting classical multi-processing, and destress mode, supporting lifetime extension of the system. In order to reduce the overhead imposed by cell level hardening, it is frequently required to utilize selective cell hardening. For this the specific, netlist level flow has been investigated and enabled, identifying flip-flops which could generate most critical SEUs (Single Event Upsets) leading to persistent error or error which propagates to primary outputs. This flow is based on Answer Set Programming (ASP) and shows very promising initial results [8].

Figure 10 describes the architecture of the HiRel demonstrator. The system includes 4 core RISC-V cores, extended with relevant interfaces, such as SpaceWire, CAN, MIL-STD-1553, SPI, UART, JTAG, and I2C. The control of the operating mode is performed by a framework controller that is at the same time able to operate as lock-step controller and voter. This system is currently under implementation and verification, and in the following months the chip fabrication is planned.

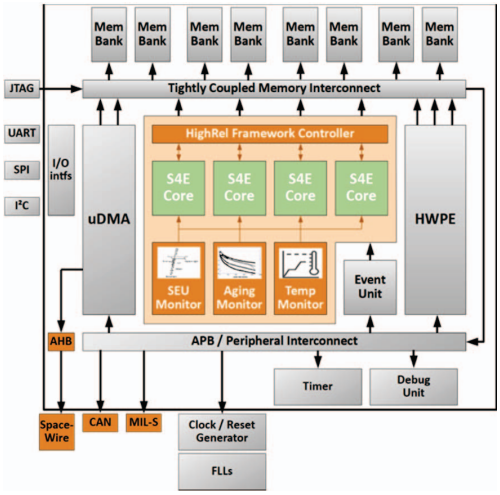

Figure 10: Architecture of HiRel demonstrator.

IV. Summary and Conclusions

This paper has introduced the concept, objectives and first results of the project Scale4Edge. The project targets at ecosystem development for scalable RISC-V based edge applications. The proposed ecosystem will be evaluated in different demonstrators. The Scale4Edge project started in Mai 2020, and first results have been already obtained and the demonstrators are under preparation. The following period will show the completion of the Scale4Edge ecosystem, and its evaluation by the demonstrators.

V. Acknowledgements

The work is supported in part by the German Federal Ministry of Education and Research (BMBF) within the project Scale4Edge under contract no. 16ME0127.

VI. References

[1] D. Mueller-Gritschneder, et al. „The Extendable Translating Instruction Set Simulator (ETISS) interlinked with an MDA Framework for fast RISC Prototyping," IEEE RSP, 2017.

[2] S. Tempel, et al., „An Effective Methodology for Integrating Concolic Testing with SystemC-based Virtual Prototypes," DATE, 2021.

[3] V. Herdt, et al. „Efficient Cross-Level Testing for Processor Verification: A RISC-V Case-Study," FDL, 2020.

[4] European Commision: „The impact of Open Source Software and Hardware on technological independence, competitiveness and innovation in the EU economy". Final Study report, 2021.

[5] J. Müller, et al. „A Formal Approach to Confidentiality Verification in SoCs at the Register Transfer Level", In 58th IEEE/ACM Design Automation Conf., 2021.

[6] P. Adelt, et al. „Fast Dynamic Fault Injection for Virtual Microcontroller Platforms". In: Proceedings of the IEEE/IFIP VLSI-SOC, Tallin, Estonia, 2016.

[7] P. Palomero Bernardo, et al. "UltraTrail: A Configurable Ultralow-power TC-ResNet AI Accelerator for Efficient Keyword Spotting." *IEEE Trans. CAD* 39.11 (2020): 4240-4251.

[8] A. Breitenreiter et al. "Reliability Analysis in Less than 200 Lines of Code." *IEEE LASCAS*, 2021.