2024 IEEE Nordic Circuits and Systems Conference (NorCAS) | 979-8-3315-1766-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/NorCAS64408.2024.10752442

AMPER-X: Adaptive Mixed-Precision RISC-V Core for Embedded Applications

Ahmad Othman, Ahmed Kamaleldin, Diana Göhringer
Technische Universität Dresden, Chair of Adaptive Dynamic Systems, Germany
E-mails: {ahmad.othman, ahmed.kamal, diana.goehringer}@tu-dresden.de

***Abstract*—The rapid expansion of the Internet of Things, wireless sensor networks, and robotics applications have established embedded systems as a fundamental component of modern computing. Despite this growth, optimizing power consumption and computing resources remains critical. Traditional embedded platforms often struggle with fixed hardware configurations that limit their adaptability and efficiency. To address these issues, this paper presents a novel adaptive processor architecture based on RISC-V ISA which introduces reconfigurable execution regions capable of supporting mixed-precision operations (16/8/4-bit). These reconfigurable execution regions execute and continuously adapt to varying precision and operational demands triggered by custom RISC-V instructions that activate the desired region at specific precision levels. The reconfiguration process is managed through dynamic function exchange (DFX) targeting AMD/Xilinx XCZU7EV FPGA device. The evaluation demonstrates that the modified core efficiently utilizes FPGA resources, with 6,396 LUTs and 2,679 FFs allocated, and operates with a dynamic power consumption of 24 mW at 50 MHz. The reconfiguration time is 76 ms for each reconfiguration execution region, incorporating the overhead required for reconfiguration management and control. These results highlight the core's potential to allow the realization of RISC-V-based reconfigurable cores without degrading performance and energy efficiency for embedded applications.**

***Index Terms*—Embedded Systems, RISC-V, Processor Architecture, Dynamic Function Exchange, FPGAs.**

I. Introduction

Nowadays, several application domains are running on embedded platforms, which necessitates further optimizations to cope with the constrained power consumption and limited computing resources of embedded platforms. Therefore, adaptive computing architectures can draw the course to support a wide range of applications by enhancing the degree of utilization and flexibility of target embedded platforms [1]. Those architectures can be adjusted to varying computational demands at runtime, optimizing resources and energy efficiency across different workloads.

In this context, several adaptable embedded platforms have been proposed by extending their general-purpose computing cores with a reconfigurable fabric [2], [3]. This aims to tackle the limited resources bottleneck through runtime reconfiguration by offloading selected compute kernels to the reconfigurable fabric as hardware accelerator modules. However, the size of the reconfigurable fabric should be large enough to host the largest hardware modules at runtime, leading to a high reconfiguration time for large workloads [4]. Accordingly, embedding adaptability within the processing core pipelines offers a promising solution for reducing hardware resource overhead and increasing programmability. By shifting reconfiguration granularity from the kernel level to the arithmetic or operation level within the core pipeline, this approach minimizes the need for extensive hardware resources for reconfigurable regions. As a result, it facilitates the development of instruction-based reconfigurable cores that can be tailored for both general-purpose and domain-specific applications [5].

With the proliferation of open-source RISC-V ISA [6] and related ecosystems, there is a growing momentum to customize the ISA to support different functionalities with different precision as well as pipeline adaptation with less development cost. From the application perspective, several application domains (e.g., deep neural network (DNN), signal processing) are increasingly optimized through quantization techniques to reduce their workload size, making them suitable for deployment on low-power embedded platforms [7]. For example, quantized neural networks (QNNs) can contain many precision formats among the layers in order to achieve the highest possible accuracy with less memory footprint. This leads to the necessity to develop an embedded processor capable of supporting mixed-precision operations, optimizing the balance between accuracy, performance, and energy efficiency across diverse workloads [8].

In this work, we propose a novel adaptive embedded core that builds upon the open-source 32-bit CV32E40P core [9], extending its capabilities with new reconfigurable execution regions (RERs). These RERs are designed to support a variety of operations with mixed precisions, enhancing the core's flexibility and efficiency. Each RER is associated with a unique custom instruction to manage and control the hosted logic or the reconfigurable module (RM). These custom instructions can support up to three input operands for each region, allowing complex operations as well as specifying the precision of the hosted logic. The core adaptability is managed and controlled through dynamic partial reconfiguration (DPR) now called dynamic function exchange (DFX) [10] to reconfigure the functionality of the RERs at runtime. The contributions of this work are delineated as follows:

- An adaptive RISC-V core based on an existing 4-stage in-order 32-bit pipeline suitable for embedded applications. The adaptive core supports up to three RERs suitable to host mixed-precision (16/8/4-bit) operations.
- This work extends the RISC-V ISA with three custom

instructions to control and manage the hosted operations for each RER and specify each region's precision. Additionally, a fourth custom instruction is added to detect which RMs are hosted by all RERs at runtime.
- The runtime adaptability to change the functionality of the RER is managed by the DFX technique targeting AMD/Xilinx FPGA devices

The rest of the paper is structured as follows: Section II discusses related work. The proposed adaptive RISC-V core and its components are introduced in Section III. Evaluation and experimental results are presented and analyzed in Section IV. Finally, Section V concludes this work.

II. Related Work

In recent decades, reconfigurability has seen notable development, becoming an increasingly prominent concept in hardware design. Dynamic reconfiguration and partial reconfiguration (PR) form the core innovations in this evolution. Dynamic reconfiguration enables the alteration of the hardware at runtime. Partial reconfiguration, on the other hand, changes parts of the hardware without impacting the entire system. Partial reconfiguration is used in a diverse range of applications, such as adaptability, overhead reduction, reliability improvement, and hardware computing [1].

This section presents related work on two main topics: reconfiguration within the pipeline of in-order processors and mixed-precision computing. The existing research in each area is explored to establish the motivation for this work and create a bridge between mixed-precision and reconfigurability. This connection is crucial for enabling adaptation in mixed-precision domain-specific applications, demonstrating how reconfigurable hardware can be utilized to optimize performance and energy efficiency.

A. Reconfigurable cores

The work on runtime-adaptive processors for embedded systems has seen significant developments across different architectures. Henkel et al. [11] introduced i-Core, a runtime adaptive processor designed for dynamic reconfiguration to optimize performance in embedded systems. Embedded in a heterogeneous, loosely coupled multi-core system, the processor leverages high memory bandwidth to expedite special instructions (SIs). It utilizes a fine-grained reconfigurable fabric for adapting pipeline stages, branch prediction, and cache/scratchpad configurations.

The i-core employs a modular approach where SIs consist of multiple data paths (DPs) dynamically loaded onto the fabric. The adaptation process begins with setting microarchitectural parameters and allocating fabric based on speedup needs. DPs are loaded in parallel to task execution, with the processor's runtime system determining the sequence and placement of DPs to optimize performance as resources become available dynamically.

In a similar way, Scheipel et al. [12] developed moreMCU, a hardware/software co-designed platform that focuses on dynamic partial reconfiguration. Unlike i-Core, which reconfigures DPs to execute its SIs, moreMCU emphasizes the adaptability of its execution stage without relying on modularity, reconfiguring an entire module at a time through its hardware/software co-design approach. This platform integrates an operating system with reconfigurable peripherals and execution units, featuring a programmable decoder that enables real-time reconfiguration of execution operations.

FlexBex [13] supports a different approach by integrating an embedded FPGA (eFPGA) with multiple execution slots into the lowRISC Ibex RISC-V core to support dynamic partial reconfiguration. The eFPGA, tightly coupled with the Ibex core, enables custom instruction execution and runtime slot reconfiguration. This integration allows the adaptation of the instruction set and the configuring of the eFPGA to meet the varying requirements of different applications. Table I summarizes adaptive-processors-related work in comparison to this work.

B. Mixed precision

Garofalo et al. [7] designed a microarchitectural extension for RISC-V cores targeting software programmable QNN inference (XpulpV2). The core was introduced with new instructions and hardware units to optimize low-precision QNNs. By leveraging a dot-product unit capable of handling sub-byte vector operations and a quantization unit to accelerate data compression, this approach increased the performance and energy efficiency of QNN workloads. Several studies [8], [14]

TABLE I: Summary of related work and proposed work

Related Work	Adaptability	Targeted Core	ISA	Freq.	Number of Reconfigurable Execution Areas	Reconfiguration Technique
i-core [11]	Pipeline Stages Branch Prediction Cache/Scratchpad	LEON3	SPARC V8	Not specified	Not specified	Not specified
moreMCU [12]	Execution Stage Operation On-Chip Peripherals	CV32E40P	RISC-V	50 MHz	3	DPR
FlexBex [13]	Execution Stage Operation	Ibex	RISC-V	Not specified	3	eFPGA
AMPER-X	Execution Stage Operation Execution Stages Mixed-Precision	CV32E40P	RISC-V	50 MHz	3	DPR

have leveraged the XpulpV2 extension to enhance RISC-V processors for various applications.

Highlighting the flexibility of mixed precision techniques, Armeniakos et al. [15] developed a configurable mixed-precision architecture based on the Ibex core. Their work introduced a SIMD (Single Instruction, Multiple Data) execution unit for mixed-precision operations. The primary focus of these modifications was to extend the RISC-V core with minimal hardware overhead, enabling the core to perform parallel operations at varying precision levels.

The contribution of our proposed adaptive core to the existing related work can be summarized in three main points. First, it introduces modular reconfigurable execution regions that manage different numbers of inputs (from 1 to 3) and support mixed precision (4-bit, 8-bit, and 16-bit). Second, it offers the flexibility to reconfigure RERs to execute any operation that meets the core execution stage design constraints at runtime through the DFX technique, with the precision level also being reconfigurable at runtime. Third, it features a self-aware reconfiguration that can adapt to execute new operations and detect the region configuration state of the region.

III. Adaptive RISC-V Core Architecture

This section discusses the base core used in this work, along with the modifications and extensions implemented to support adaptive fine-grained mixed-precision operations. Also, design choices and enhancements are discussed with a focus on their adaptability to varying precision requirements. In this work, the CV32E40P core [9] is used as the base RISC-V core for the proposed adaptive RISC-V core. It is based on a 32-bit four-stage pipeline supporting RV32 ISA. Furthermore, for testing and evaluation, the adaptive RISC-V core is integrated into a bus-based RISC-V SoC architecture proposed by [16]. The CV32E40P serves as the base pipeline architecture for the current work, which introduces architectural enhancements aimed to establish an adaptive core. Specifically, the core has been modified to integrate three RERs, each designed to accommodate arithmetic units with up to three inputs. Mixed-precision adaptation used in this work increases the core's flexibility by allowing it to reconfigure its execution stage at runtime to adjust precision based on application requirements.

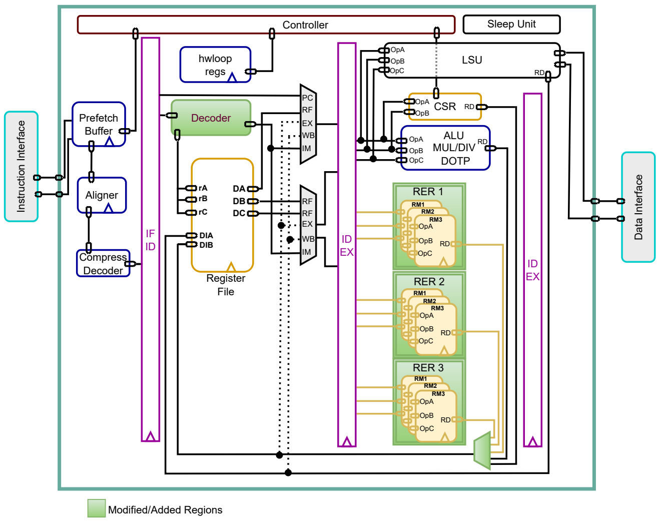

Fig. 1: The CV32E40P [9] core modifications.

A. Core Modifications for Mixed Precision Adaptation

The proposed core is designed to dynamically adapt both precision and functionality. Each region can be reconfigured at runtime to handle multiple functions and precision levels, based on the specific requirements of the application at hand.

- Precision Adaptation: The core has been modified to execute multiple precision levels, with the capability to control the region precision and adapt it dynamically at runtime. For tasks that demand high-speed processing with low accuracy, the regions are configured to operate at low precision to boost throughput and reduce power consumption. Meanwhile, for tasks that require high computational accuracy higher precision configuration is used.
- Functional Adaptation: Each region within the core is designed to execute any function that fits within its RTL (Register Transfer Level) design constraints, accommodating operations with one to three inputs. This flexibility includes support for both single-cycle and multi-cycle functions, allowing the core to efficiently handle a broad range of computational tasks.

Fig. 1 illustrates the modifications made to the CV32E40P core to support the three RERs in the execution stage. Each region can perform one specific function with a designated precision.

- Instruction Fetch and Decode: Instructions are fetched and passed to the decoder through the IF/ID pipeline stage. The decoder then generates enable and precision select signals specific to each region, based on the received instruction. Each instruction triggers the corresponding region, allowing it to execute a specified arithmetic operation with the selected precision. The connection between RER and decoder is detailed in Fig. 2.
- Execution Stage: In the execution stage, the inputs (rs1, rs2, rs3) are sent to all regions along with their respective enable signals. Only one enable signal is activated at a time, which selects the appropriate region for processing. A multiplexer (MUX) then chooses the result from the active region and sends it back to the register destination as shown in Fig. 2.
- Stalling Mechanism: The processor stalls if no arithmetic unit is configured in the selected region or if the region is still processing a multi-cycle operation. The ready signal as shown in Fig. 2 controls the core stalling. This mechanism ensures that the core does not proceed with subsequent instructions until the current operation is complete.
- New Instruction and Tag Retrieval: To enhance the functionality of the reconfigurable core, a new instruction

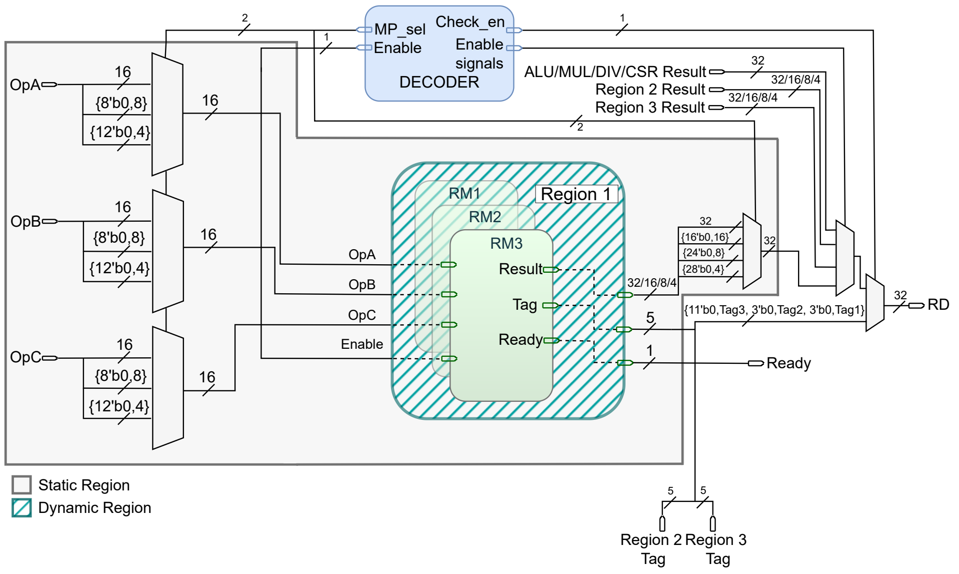

Fig. 2: A single reconfigurable execution region, including part of the writeback logic.

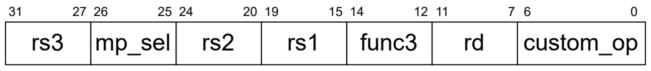

Fig. 3: The used RISC-V custom instruction format.

has been implemented. This instruction enables users to retrieve the configuration status of each region, providing information on the current function and precision level in use. It also indicates whether a region is currently not configured. This enables efficient monitoring and management of the core's operational state, ensuring configuration control and making the core self-aware.

B. *Custom Instructions for Reconfigurable Execution Regions and Mixed-Precision Support*

The standard RISC-V "I" extension has been augmented to include custom instructions designed for executing and monitoring reconfigurable regions operations, supporting all precision levels and the tag retrieval instruction. Fig. 3 illustrates the used instruction format.

- mp_sel field: The mp_sel field controls the precision of the region, allowing dynamic switching between different precision levels.
- func3 field: The func3 field is designed to support the configuration of multiple units within the same region. However, this field is not utilized in this paper as only one function is implemented per region.
- Opcode field: The opcode is responsible for identifying and selecting the specific region to be activated. Each region has a unique opcode, ensuring that each operation executes in the correct region.
- Register fields: The fields rs1, rs2, rs3, and rd denote the addresses of the source and destination registers. These fields specify the operands for the arithmetic operation and the register address where the result will be written back. Each region can use up to three source registers, but it is also capable of using fewer, depending on the region's reconfiguration.

This augmentation of the RISC-V ISA provides a robust mechanism for controlling the core's reconfigurable regions, enabling dynamic adaptation capabilities. Table II demonstrates the adaptive mixed-precision core instruction extension encoding.

TABLE II: Mixed-precision ISA extension encoding

Custom Insr.	rs3	mp_sel	rs2	rs1	rd	Function
cust_X_4	4-bit Acc/ Source	00	4-bit	4-bit	4-bit	4-bit operations
cust_X_8	8-bit Acc/ Source	01	8-bit	8-bit	8-bit	8-bit operations
cust_X_16	16-bit Acc/ Source	10	16-bit	16-bit	16-bit	16-bit operations
cust_X_16U	32-bit Acc/ Source	11	16-bit	16-bit	32-bit	Regions 32-bit result
check_con	-	-	-	-	Tags	Monitor the regions

X: represents the number of the RER.

C. *Reconfigurable Execution Region*

Fig. 2 illustrates the detailed layout of the reconfigurable regions and their interconnection with the decoder and core. This diagram illustrates how RM inputs and outputs are managed within the modified CV32E40P core to ensure seamless adaptability. Each of the three reconfigurable regions is designed to align with the overall core architecture and design constraints. Each region receives three inputs, along with an enable signal to initiate the operation, and generates three distinct outputs: the result of the arithmetic operation, a tag that indicates the RM's current configuration (including the function and precision in use), and a ready signal that indicates when the computation is complete and the result is available.

- Input Masking: The least-significant 16 bits of the three input registers (rs1, rs2, and rs3) are passed through a multiplexer (MUX), which adjusts the data width to match the selected precision using the control signal mp_sel. This ensures that the input data is masked appropriately for the arithmetic operation to be performed in the configured precision for that region.
- Output Handling: In the same manner, the output from each region is also passed through a MUX that selects the appropriate precision to be sent to the destination register. For operations requiring extended precision when the result can be more than 16-bits, such as multiplication, the result is extended to 32 bits to accommodate the larger output size.
- Tag Handling: Each region generates a tag representing its current configuration. These tags are concatenated together and routed through a MUX controlled by "Check_en" signal, which then selects the concatenated tags to be written to the destination register instead of the execution result as shown in Fig. 2. This comprehensive status word provides the software with vital information about the precision levels configured in each region and their respective functions. This tagging mechanism enables the software to monitor and manage the core's operational state effectively.

By incorporating these design constraints and elements, the reconfigurable regions in the proposed core achieve a high degree of flexibility, allowing the processor to efficiently adapt to a wide range of computational tasks. By leveraging a novel tag mechanism, the core attains self-aware adaptability, dynamically controlling reconfiguration at runtime.

IV. Evaluation

The proposed adaptive RISC-V core has been evaluated by integrating it into a RISC-V SoC architecture [16] targeting the AMD/Xilinx Ultrascale+ MPSOC XCZU7EV. As shown in Fig. 4, the SoC includes the proposed adaptive RISC-V core connected to instruction and data memories through an AXI interconnect. The SoC is connected to the processing system (PS) through Xilinx AXI FIFO and direct memory access (DMA) IPs for communication and synchronization during the reconfiguration process.

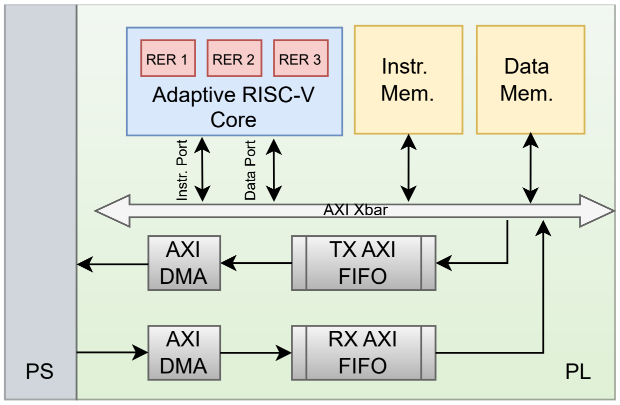

Fig. 4: A schematic of the RISC-V SoC including the adaptive RISC-V core and the connectivity to the PS.

In this work, the PS is responsible for managing the reconfiguration process over the Pynq framework [17] by loading the corresponding partial bitstream for each RM from off-chip memory to the FPGA configuration memory. Vivado ML 2023.2 is used for RTL synthesis, simulation, place and routing, and partial bitstream generation. The adaptive RISC-V core and SoC peripherals run at a clock frequency of 50 MHz. This section evaluates the proposed adaptive RISC-V core based on hardware resource utilization and the reconfiguration cost in terms of hardware overhead and reconfiguration time.

A. Hardware Resource Evaluation and Prototyping

Table III presents the detailed hardware resource evaluation of the adaptive RISC-V core and the SoC components implemented on the programmable logic (PL) side of the Zynq Ultrascale+ MPSoC as shown in Fig. 4. The adaptive core consumes $\sim$ 14% LUTs and $\sim$ 18.5 % FFs more than the base CV32E40P core due to the conducted modifications to support three RERs and the related adjustments in the decode stage to execute the new custom instructions. For prototyping and evaluation purposes, the adaptive core is connected to instruction and data memories in order to execute several software kernels for testing and verification. The instruction memory stores the software binary files generated after software compilation using the RISC-V GNU toolchain [18] where the data memory acts as the local scratchpad memory of the adaptive core. Additionally, the adaptive core is linked to the processing system (PS) side through two AXI FIFOs, facilitating data transmission and receiving during the reconfiguration process. The adaptive core's dynamic power consumption, estimated at 24 mW using Vivado power estimator, makes it suitable for low-power embedded platforms. A fixed reconfigurable partition size, as listed in Table III, is assigned to the three RERs for hosting several mixed-precision RMs at runtime.

B. Reconfiguration Cost

In this work, the reconfiguration process is managed by both the adaptive RISC-V core and the PS. The adaptive core initiates the reconfiguration process by first checking the tags of hosted RMs by all RERs to determine whether a new RM for a specific operation/precision requested by the software is necessary to be deployed. In case the requested operation/precision is not available on the RERs, the adaptive core sends a request message to the PS via the TX AXI-FIFO including the tag of the RER and the specific RM for this RER. Next, the PS starts the DFX process by loading the specific partial bitstream corresponding to the selected RM from the off-chip memory to the target RER. After the DFX process is done, the PS sends a message via the RX AXI-FIFO to the

TABLE III: Total resource utilization and estimated power consumption targeting AMD/Xilinx XCZU7EV FPGA.

Resource \ Units	SoC	Adaptive / CV32E40P Core	Instr. Mem	Data Mem.	AXI Xbar and DMAs	RER 1	RER 2	RER 3
LUTs	15209	6396 / 5640	410	406	5261	912	912	912
FFs	15241	2679 / 2262	290	339	6173	1920	1920	1920
BRAMs	84	0 / 0	15	64	5	0	0	0
DSPs	7	7 / 7	0	0	0	0	0	0
Power (dynamic)	**24 mW** (consumed by the modified RISC-V core at Freq = 50 MHz)							

TABLE IV: Reconfigurable modules resource utilization and reconfiguration time of a single RER.

Resource \ RMs	MAC 16	MAC 8	MAC 4	MUL 16	MUL 8	MUL 4	Sin 16	Sin 8	Sin 4	Cos16	Cos 8	Cos 4
LUTs	249	114	28	224	93	25	537	107	65	453	104	51
FFs	0	0	0	0	0	0	81	20	16	79	20	0
RER Utilization (%)	27.3	12.5	3	24.6	10.2	2.74	58.9	11.73	7.12	49.7	11.4	5.5
Reconfiguration Time*	**76 ms** (It includes the software overhead required by the RISC-V and the PS to manage the reconfiguration process)											

*through PS-PCAP interface, PCAP: Processor Configuration Access Port.

adaptive core carrying a reconfiguration process "done" signal indicating the availability of the selected RM. Afterwards, the adaptive core starts executing the corresponding custom instruction to this RER with the new hosted operation/precision.

Fig. 5 shows the complete FPGA floorplan of the RISC-V SoC, including the adaptive core with three RERs. The adaptive core consumes $\sim$ 50% of the SoC resources. The RER sizes are selected to meet the resource requirement of the largest RM as well as the DFX constraints regarding partition size. As shown in Table IV, several mixed-precision operations are implemented as RMs for testing and verification of the proposed adaptive core. The RMs support different types of operations with three, two, and one input operands. Also, their RER utilizations based on the selected precision are shown in Table IV. The reconfiguration time is measured by the adaptive RISC-V core clock counter register including the software overhead of the reconfiguration management process handled by both the RISC-V core and the PS. As shown in Table IV, for a single RER, the achieved reconfiguration time is 76 ms.

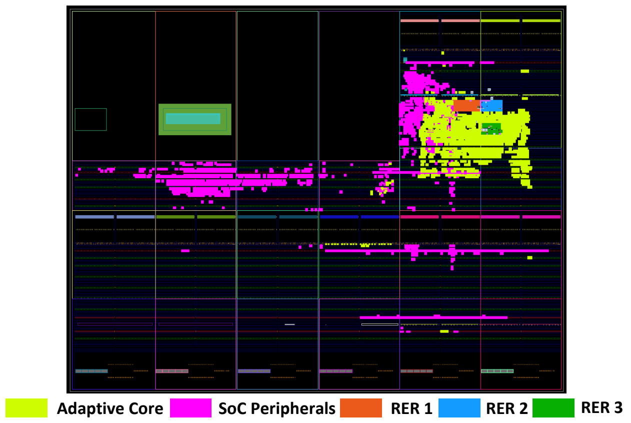

Fig. 5: The FPGA floorplan on the XCZU7EV shows the complete RISC-V SoC with the three RERs within the adaptive RISC-V core.

C. *Re-utilization and Efficiency*

The adaptive RISC-V core enhances efficiency by dynamically adjusting operational precision and resource allocation, optimizing resource usage, and minimizing computational overhead. Lower precision operations (e.g., 4-bit) consume less power and resources than higher precision ones (e.g., 16-bit), potentially leading to energy savings, especially in applications like image processing and machine learning. Improving core adaptability to better fit real-world applications involves reducing the reconfiguration time, which currently stands at 76 ms per RER due to external memory accesses for partial bit storing/loading. Implementing a "configuration cache" to store frequently used configurations locally would enable faster switching between precision levels without external memory access.

V. Conclusion

This paper introduces a novel adaptive RISC-V core architecture featuring multiple RERs that support mixed-precision operations, significantly enhancing flexibility and efficiency. By integrating RERs, the architecture enables runtime dynamic partial reconfiguration, supporting multiple integer precision and operational capabilities to meet diverse application needs. Evaluation on an AMD/Xilinx FPGA platform demonstrates that the core efficiently utilizes resources, achieving a power consumption of 24 mW at 50 MHz with a reconfiguration time of 76 ms per RER. The design minimizes hardware overhead while maximizing programmability, offering a versatile and efficient solution for embedded systems, with precision adaptable to specific application needs.

VI. Acknowledgment

This work has been partially funded by the German Federal Ministry of Education and Research BMBF as part of the DAKORE (Datenfunknetz mit Adaptivhardware und KI-Optimierung zur Reduktion des Energieverbrauchs) project under grant agreement number 16ME0433K and by the German Research Foundation (“Deutsche Forschungsgemeinschaft”) (DFG) under Project-ID 287022738 TRR 196 for Project S05.

REFERENCES

[1] K. Vipin and S. A. Fahmy, “FPGA Dynamic and Partial Reconfiguration,” in ACM Computing Surveys, vol. 51, no. 4, pp. 1–39, Jul. 2018.

[2] N. Charaf, A. Kamaleldin, M. Thümmler and D. Göhringer, "RV-CAP: Enabling Dynamic Partial Reconfiguration for FPGA-Based RISC-V System-on-Chip," in IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Portland, OR, USA, 2021, pp. 172-179.

[3] P. D. Schiavone et al., “Arnold: an eFPGA-Augmented RISC-V SOC for flexible and Low-Power IoT end nodes,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 4, pp. 677–690, Apr. 2021.

[4] A. Kamaleldin, I. Ahmed, A. M. Obeid, A. Shalash, Y. Ismail and H. Mostafa, "A Cost-Effective Dynamic Partial Reconfiguration Implementation Flow for Xilinx FPGA," in New Generation of CAS (NGCAS), Genova, 2017, pp. 281-284.

[5] A. Chattopadhyay, “Ingredients of Adaptability: A Survey of Reconfigurable Processors,” in VLSI Design, vol. 2013, pp. 1–18, Jul. 2013.

[6] A. Waterman, Y. Lee, D. Patterson, and K. Asanović, “The RISC-V Instruction Set Manual, Volume I: User- Level ISA, Version 2.1,” 2016. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-118.pdf

[7] A. Garofalo, G. Tagliavini, F. Conti, D. Rossi and L. Benini, "XpulpNN: Accelerating Quantized Neural Networks on RISC-V Processors Through ISA Extensions," in Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020, pp. 186-191

[8] N. Bruschi, A. Garofalo, F. Conti, G. Tagliavini, and D. Rossi, “Enabling mixed-precision quantized neural networks in extreme-edge devices,” in Proceedings of the 17th ACM International Conference on Computing Frontiers (CF ’20). Association for Computing Machinery, USA, 2020, pp. 217–220.

[9] OpenHW Group CORE-V CV32E40P RISC-V IP, open-source hardware project, https://docs.openhwgroup.org/projects/cv32e40p-usermanual/en/latest/index.html.

[10] Xilinx Inc., Vivado Design Suite User Guide: Dynamic Function eXchange, UG909 (v2023.2), Nov. 15, 2023. [Online]. Available: https://docs.amd.com/viewer/book-attachment/sBScSL3FKc03Va0VGNQb4w/6bGHmfY3ygosIBsuyij23Q.

[11] M. Damschen, M. Rapp, L. Bauer, and J. Henkel, “i-Core: A Runtime-Reconfigurable Processor Platform for Cyber-Physical Systems,” Springer eBooks, 2020, pp. 1–36.

[12] T. Scheipel, F. Angermair and M. Baunach, "moreMCU: A Runtime-reconfigurable RISC-V Platform for Sustainable Embedded Systems," in 25th Euromicro Conference on Digital System Design (DSD), Maspalomas, Spain, 2022, pp. 24-31.

[13] N. Dao, A. Attwood, B. Healy and D. Koch, "FlexBex: A RISC-V with a Reconfigurable Instruction Extension," in International Conference on Field-Programmable Technology (ICFPT), Maui, HI, USA, 2020, pp. 190-195.

[14] G. Ottavi, A. Garofalo, G. Tagliavini, F. Conti, L. Benini and D. Rossi, "A Mixed-Precision RISC-V Processor for Extreme-Edge DNN Inference," in IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Limassol, Cyprus, 2020, pp. 512-517.

[15] G. Armeniakos, A. Maras, and D. Soudris, “Mixed-precision Neural Networks on RISC-V Cores: ISA extensions for Multi-Pumped Soft SIMD Operations,” 2024. Accessed: Aug. 14, 2024. [Online]. Available: https://arxiv.org/pdf/2407.14274

[16] A. Kamaleldin, M. Ali, P. Amini Rad, M. Gottschalk and D. Göhringer, "Modular Memory System for RISC-V Based MPSoCs on Xilinx FPGAs," in IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Singapore, 2019, pp. 68-73.

[17] PYNQ: Python productivity for Adaptive Computing Platforms. [Online]. Available: https://pynq.readthedocs.io/en/latest/index.html

[18] RISC-V GNU Compiler Toolchain: https://github.com/riscv-collab/riscv-gnu-toolchain