FPGA-Accelerated Simulation of a RISC-V Vector Architecture

Kathryn Chapman
Department of Computer Science
Lawerence Livermore National Laboratory, National TsingHua University
Hsinchu, Taiwan
kjchapman@pllab.cs.nthu.edu.tw

Maya Gokhale
Center for Applied Scientific Computing
Lawerence Livermore National Laboratory
Livermore, USA
gokhale2@llnl.gov

Hiren Patel
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
hiren.patel@uwaterloo.ca

Jenq-Kuen Lee
Department of Computer Science
National TsingHua University
Hsinchu, Taiwan
jklee@cs.nthu.edu.tw

***Abstract*—A profusion of RISC-V based System on Chip (SoC) accelerator architectures have emerged in the past couple of years, fueled by AI and high performance computing. In many of these architectures, the RISC-V CPU serves as a control processor to route data and instructions to specialized IP blocks. Design, implementation, optimization, and validation of such architectures ultimately requires modeling and simulation of the entire integrated SoC at the Register Transfer Level (RTL) in software simulators (for visibility into design internals) and FPGA (for fast simulation of realistic workloads). In this work, we target a RISC-V scalar core augmented with a high performance ARA2 configurable floating-point vector unit on workloads that are particularly challenging for software RTL simulation. To accelerate simulation, we map the RTL design to the AMD ZCU102 FPGA development board. On this platform, we implement a heterogeneous dual System on Chip in which an ARM CPU core on the Zynq UltraScale+ manages benchmark setup, execution, and result collection for the RISC-V system, which is implemented in the Programmable Logic. The dual SoC implementation enables fast, autonomous performance evaluation of the ARA2 design.**

***Index Terms*—RISC-V, FPGAs, Vector Processor, Heterogeneous Systems, Shared Memory**

I. Introduction

The ARA2 RISC-V Vector architecture [1] consists of a CVA6 RISC-V scalar core (CVA6) augmented by a design-time configurable vector unit. The open source design [2] provides RTL for the entire SoC as well as a port to an FPGA. In this work, we map the FPGA design to a previously unsupported FPGA, the AMD Zynq UltraScale+ on a ZCU102 development board. The ARA2 SoC (CVA6 and ARA2 vector unit) is instantiated on the Programmable Logic, and runs applications using its dedicated memory under the control of an ARM core on the Processing System of the Zynq UltraScale+. Applications and directives on data generation are pre-loaded on an SD card and are transferred by the ARM core to the ARA2 SoC. Results generated by ARA2 are transferred by the ARM to the filesystem, available for offline analysis. Unlike RTL simulation and the FireSim-based FPGA implementation of ARA2 presented in [3], our implementation fully supports external memory. Consequently, our system can run large workloads with data residing in DRAM.

II. Design and Implementation

Our FPGA implementation for the ZCU102 augments the FPGA port developed by the ARA2 team for the AMD VCU128, a $15,000 board. The ZCU102 is a widely available, mature, and more affordable ($3,234) alternative with the added benefit of supporting heterogeneous systems with two varieties of ARM cores in addition to soft processors instantiated in the Programmable Logic (PL), specifically the CVA6 RISC-V CPU with ARA2 vector unit (called Cheshire below to reflect nomenclature from the ARA2 tool flow). In contrast to the prior ARA2 FPGA implementation on VCU128, we exploit ARM processors on the ZCU102's Zynq UltraScale+ FPGA to control Cheshire.

As shown in Fig. 1, the Zynq UltraScale+ is composed of two parts, "PS-side" containing the ARM complex, its dedicated DRAM , and connection to I/O peripherals, and "PL-side" containing the Cheshire SoC plus our custom logic. In addition to Cheshire, we have designed memory addressing and accessing logic to manage a separate PL-side DRAM, and debug/reset circuitry.

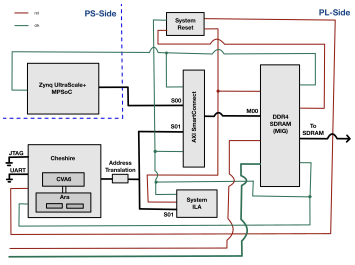


Fig. 1. ARM PS-side and Cheshire+Memory PL-side

The Cheshire accesses the PL-side DDR4 memory interface module for instruction and data, which are provided by the ARM. The AXI SmartConnect module from the Xilinx IP library arbitrates between ARM and Cheshire: ARM writes instruction and data to the memory and reads back results; Cheshire reads and writes the memory to execute the simulation. The SmartConnect communicates with the DDR4 SDRAM Memory Interface Generator (MIG) module that accesses the PL-side DRAM.

Since CVA6 expects memory addresses to begin at a specific address 0x8000_0000, which is also reserved by the ARM, we add a small address translation unit to relocate CVA6 addresses to 0x4_8000_0000, which eliminates the need to modify Cheshire internals. The address remapping is shown in Fig. 2.

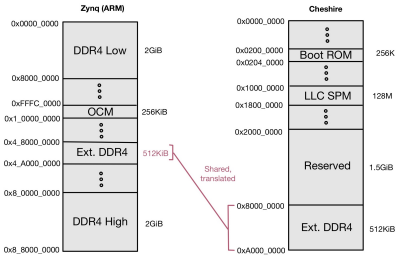


Fig. 2. Memory Map for Cheshire Address Translation

Our framework supports simulating a suite of benchmarks/applications in a single autonomous session. In our design, an ARM core sets up simulation experiments to run in Cheshire. Array data is not transferred over JTAG. Instead, it is initialized by the ARM, eliminating transfers over the 10 MHz JTAG interface and offloading much of the workload from the 50 MHz Cheshire SoC. The ARM also operates at 1.2 GHz, enabling significantly faster data initialization.

Coordination between ARM and Cheshire is through mailbox style delivery and polling at pre-defined DRAM locations. The ARM loads the simulation workload including program, data, and metadata from SD card into DRAM, initiates the CVA6 bootrom program, monitors a pre-defined DRAM location for completion, and reads the results from DRAM to a file on SD card.

Creating a workload involves both RISC-V and ARM programs. Our software toolchain expands the ARA2 Cheshire workflow by coordinating data structures used in the simulation workload with corresponding data structures in the ARM program. The software flow is used to build the workload we have evaluated, a subset of the TSVC benchmarks [4], which contain specifically designed vector loops.

III. Evaluation

Evaluation of ARA2 focuses on effective use of the vector unit to accelerate performance. The official software testing and benchmark suite for ARA2 primarily consists of assembly and intrinsic-based implementations, leaving the role of the compiler largely unexplored. We therefore evaluate compiler autovectorization using representative benchmarks (Table I) from the TSVC suite. The benchmarks can be configured to multiple problem sizes, from tiny to huge. Due to slow RTL software simulation, we could only run tiny and small problem sizes. On the ZCU102, we can run larger problem sizes, with results generated in seconds. We can also run multiple iterations and select array datatypes as int, float, or double. Output for each benchmark is the cycle count and number of instructions executed.

For FPGA evaluation, we report large problem size results using floating-point data (array LEN = 1,024,000). IPC values are generally stable from 1 to 20 iterations. Tiny (LEN = 15,360) and small (LEN = 30,720) checks showed similar IPC values, so only large results are included in Table II.

TABLE I
Categories of Compiler Vectorization Tests

Benchmark	Category
LINEAR DEPENDENCE	s000
INDUCTION VARIABLE	s121
GLOBAL DATA FLOW	s131
CONTROL FLOW	s162
SYMBOLICS	s171
STATEMENT REORDERING	s211 (not vectorizable)
NODE SPLITTING	s243
EXPANSION	s251
CROSSING THRESHOLDS	s1281
REDUCTIONS	s311
PACKING	s341 (not vectorizable)

IV. Discussion

In this work, the Ultrascale+ ARM CPU manages benchmark setup, execution, and result collection for the Cheshire system, enabling fast, autonomous evaluation of the ARA2 vector processor. This platform provides a flexible and efficient framework for future evaluation and development of RISC-V vector architectures. The availability of ARM cores enables additional use cases, including high-speed data generation and runtime inspection of execution results to support more autonomous simulation workflows. Future work will focus on further optimizing the memory hierarchy to better control DRAM latency to obtain more realistic memory behavior and a more representative evaluation of the system.

Acknowledgment This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 under LDRD Project 25-ERD-016.

References

[1] M. Perotti, M. Cavalcante, R. Andri, L. Cavigelli, and L. Benini, " Ara2: Exploring Single- and Multi-Core Vector Processing With an Efficient RVV 1.0 Compliant Open-Source Processor ," *IEEE Transactions on Computers*, vol. 73, no. 07, pp. 1822–1836, Jul. 2024. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TC.2024.3388896

[2] ETH and U. of Bologna, "Ara," https://github.com/pulp-platform/ara, 2026, accessed: 2026-06-03.

[3] M. Pikuła, "Accelerating software development for emerging isa extensions with cloud-based fpgas: Rvv case study," https://serenitycode.dev/post/2024/06/riscv-summit-europe/, 2024, accessed: 2026-804.

[4] LLVM, "Tsvc benchmarks," https://github.com/llvm/llvm-test-suite/tree/main/MultiSource/Benchmarks/TSVC, 2026, accessed: 2026-06-03.

TABLE II
TSVC BENCHMARK CYCLES AND INSTRUCTIONS FOR LARGE ARRAY SIZE

Test	**ntimes=1**		**ntimes=20**		**instret/cycles**	
	cycles	**instret**	**cycles**	**instret**	**ntimes=1**	**ntimes=20**
s000	4 602 108	384 032	91 872 078	7 680 303	0.0834	0.0836
s121	9 051 565	528 050	180 967 276	10 560 563	0.0583	0.0584
s131	15 085 430	880 068	301 612 030	17 600 923	0.0583	0.0584
s162	3 003 936	208 029	59 921 651	4 160 227	0.0693	0.0694
s171	2 773 605	176 024	45 069 471	3 520 183	0.0635	0.0781
s211	60 958 452	17 407 991	1 224 656 932	348 159 484	0.2856	0.2843
s243	2 242 779	128 025	44 853 093	2 560 183	0.0571	0.0571
s251	11 923 698	896 068	238 469 818	17 920 752	0.0752	0.0751
s1281	20 493 834	1 600 097	410 037 318	32 001 009	0.0781	0.0780
s311	90 764 840	1 760 088	1 815 294 140	35 201 418	0.0194	0.0194
s341	19 800 037	9 216 021	399 364 129	184 320 141	0.4655	0.4615