Developing Custom RISC-V ISA Extensions for General Embedded Image Processing Operations

Stephan Nolting[1], Ingo Hoyer[1], Alexander Utz[1], Holger Kappert[1] and Günter Grau[2]

[1] *Fraunhofer Institute for Microelectronic Circuits and Systems (IMS), Department for Smart Sensor Systems, Duisburg, Germany*

[2] *advICo microelectronics GmbH, Recklinghausen, Germany*

Fraunhofer IMS

Fraunhofer Institute for Microelectronic Circuits and Systems

Introduction

On-site data processing (e.g. smart sensors devices)
Reduce transmission bandwidth, offload central processing system, shift real-time constraints to the device / sensor node

Application

- Feature extraction from image data using sliding-window-based processing
- General-purpose operations: standard deviation, arithmetic mean, sqrt

Challenges

Small Architecture / Foot-Print | Low Power | Performance & Flexibility

One viable solution: Application-Specific Instruction-Set Processor (ASIP)

➢ *RISC-V ISA was explicitly designed to support custom extensions [1]*

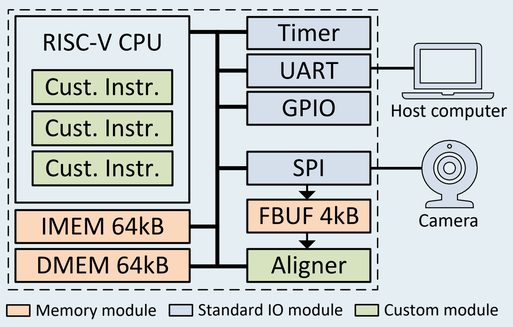
Implementation

Base Core: NEORV32 RISC-V processor [2]

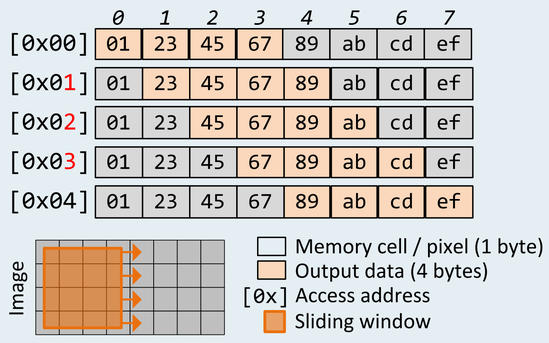
- 32-bit `rv32im_Zicsr` m/u-mode ISA; multi-cycle architecture
- Open-source with a permissive license (BSD 3-clause)
- Area-optimized; highly configurable and extensible
- Full SoC (memories and standard peripherals) and software-framework

Custom Modules

- SPI + DMA module; automatic sampling of image data (background of CPU)
- Frame buffer with *data aligner* – "in-flight" alignment of pixel data into 32-bit registers to exploit SIMD parallelism
- Three custom instructions (*so far*)



Simplified system / SoC block diagram



Concept of the data aligner

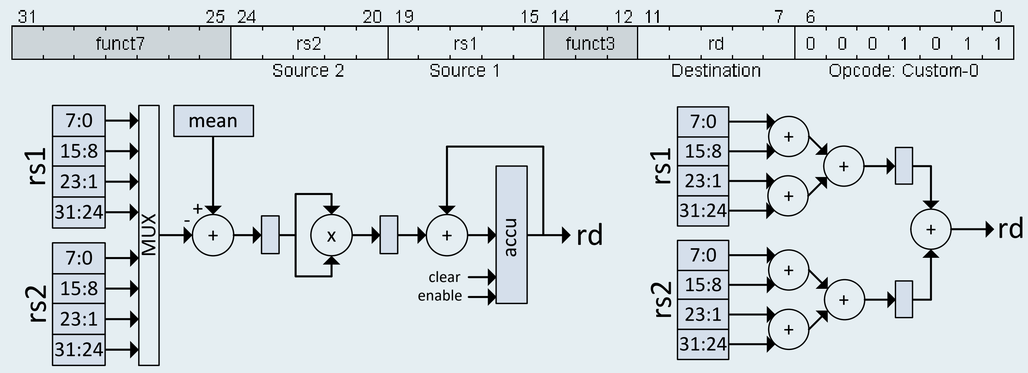
Custom Instructions

- Exploit sub-word parallelism (vertical SIMD) by custom r3-type instructions
- Vector input (8-bit pixel data; 4 pixel per 32-bit register), scalar output
- Use of `intrinsics` – no need to modify compiler / backend

➢ SIMD sum of squared differences: $rd \leftarrow \sum_{i=0}^{7}(mean - rs_i)^2$
[10 cycles latency]

➢ SIMD sum of all elements: $rd \leftarrow \sum_{i=0}^{7} rs_i$
[2 cycles latency]

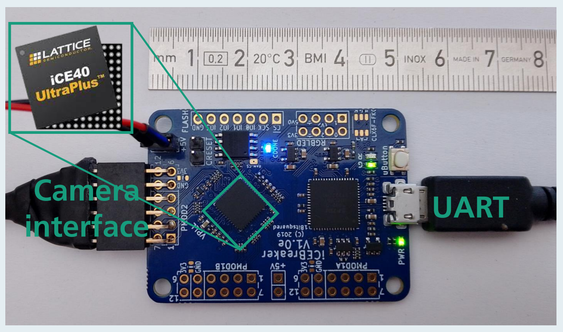
➢ Integer square root: $rd \leftarrow \sqrt[2]{rs}$ (result in 16.16 fixed-point format)
[32 cycles latency]



Custom instructions: custom r3-type instruction word layout (top); SIMD sum of squared differences architecture (left); SIMD summation architecture (right)

Evaluation

Test setup: *Icebreaker FPGA board* based on a 40 nm Lattice iCE40 Ultra Plus 5k ultra low power FPGA [3]



Lab prototyping platform

Processing Time (clock cycles @ 24 MHz)

Algorithm Part	SW-only	SW + cust. instr.
Arithmetic mean	499	77
Standard deviation	5636	158
Square root	833	43
Frames/s	**0.91**	**13.1**

x14.4

FPGA Utilization (Lattice iCE40UP5K)

Module	LUTs	FFs	BRAMs	DSPs
CPU Core	3451	1301	4	0
Cust. Instr.	530 (15%)	316	0	3

✓ **Speed-up by a factor of 14.4**
✓ **Increased resource utilization by just 15%**
✓ **No impact on the core's critical path**

Future Work

- Evaluate alternative base cores providing more computational power
  - ➢ **Fraunhofer IMS AIRISC** (`rv32imfcpx` ISA, fully-pipelined)
- Identify further computational bottlenecks (additional accelerators)
- Static and dynamic power analysis (FPGA and ASIC)
- ASIC implementation (using a low power technology)

AIRISC

The Fraunhofer IMS AIRISC Processor optimized for embedded AI [4]

Contact

Stephan Nolting
stephan.nolting@ims.fraunhofer.de
airisc@ims.fraunhofer.de
Fraunhofer IMS, Duisburg, Germany
www.ims.fraunhofer.de

1 RISC-V International, "**RISC-V Instruction Set Manual**", *github.com/riscv/riscv-isa-manual*
2 S. Nolting et al. "**The NEORV32 RISC-V Processor**", *github.com/stnolting/neorv32*, doi:10.5281/zenodo.7715920
3 Lattice Semiconductor, "**iCE40 UltraPlus ML/AI Low-Power FPGAs**", *latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus*
4 Fraunhofer IMS, "**The AIRISC RISC-V Processor for Embedded AI**", *github.com/Fraunhofer-IMS/airisc_core_complex*

RISC-V® Summit Europe 2023
5-9 June 2023
Barcelona, Spain