Leveraging the Openness and Modularity of RISC-V in Space

Stefano Di Mascio

13th ESA Workshop on Avionics, Data, Control and Software Systems (ADCSS)

TU Delft

Outline

- The RISC-V ISA
  - RISC-V in Space
- RISC-V processors
  - General Purpose processors
  - From Microcontrollers to Manycore Processors
  - Vector Processors
- Conclusion

Why the RISC-V ISA?

- The Instruction Set Architecture of a processor is **its interface between HW and SW**
  - It enable the use of certain **software ecosystem and toolchain**
    - You want an ISA already used by many people → **Large user base**
    - If you use a proprietary ISA you have to pay royalties → **Openness**
- ISAs can ‘overspecify’. For instance:
  - Design a complicated ISA to optimize certain aspects of processors
  - Functionalities not fully exploited by users or not effective for some applications
  - Implementations are more complicated than those based on simpler ISAs
- Keep the ISA **simple** and ‘**general**’, complicate the microarchitecture to meet the required level of performance
- What do we do when more complicate instructions are actually needed?
  - A simple ‘**base**’ ISA and optional ISA **extensions** with the optimized instructions that are relevant to your application → **Modularity**
- RISC-V offers **modularity**, **openness** and an **already existing user base**

Why RISC-V in Space? (1)

- Openness (not a novelty for space):
  - **Of the ISA:** SPARC V8 is also an open ISA
    - SPARC V8 based on the SPARC V7 (Sun Microsystems), RISC-V started from scratch and discussed by working groups
  - **Of the implementations:** code of LEON available to the public
- Already a relatively large user base for terrestrial applications (quite a novelty)
  - Toolchain and software ecosystem available (GCC, GDB, OSs, etc.)
  - Already (RISC-V project started in 2010, first open-source core in 2014) 50 IP cores available listed in https://riscv.org/risc-v-cores/
    - Of which 30 are open-source
    - Of which 19 are in 'industry-standard' HDLs (SystemVerilog,Verilog,VHDL)
    - Of which 4 are in VHDL
  - «Spin-in» and exploiting synergies possible
  - Potentially even more popular in the future
    - Interesting mix of academia, start-ups, and big players working on RISC-V

Why RISC-V in Space? (2)

Modularity: fit for a wide range of applications (big novelty)

- 1 Base + Standard Unprivileged Extensions + Privilege modes

Base Integer Instruction Set	
RV32I	32-bit integer
RV32E	32-bit integer (reduced)
RV64I	64-bit integer
RV128I	aka “why not planning ahead?”

Privilege modes	
M	Machine Mode (mandatory)
S	Supervisor Mode
U	User Mode

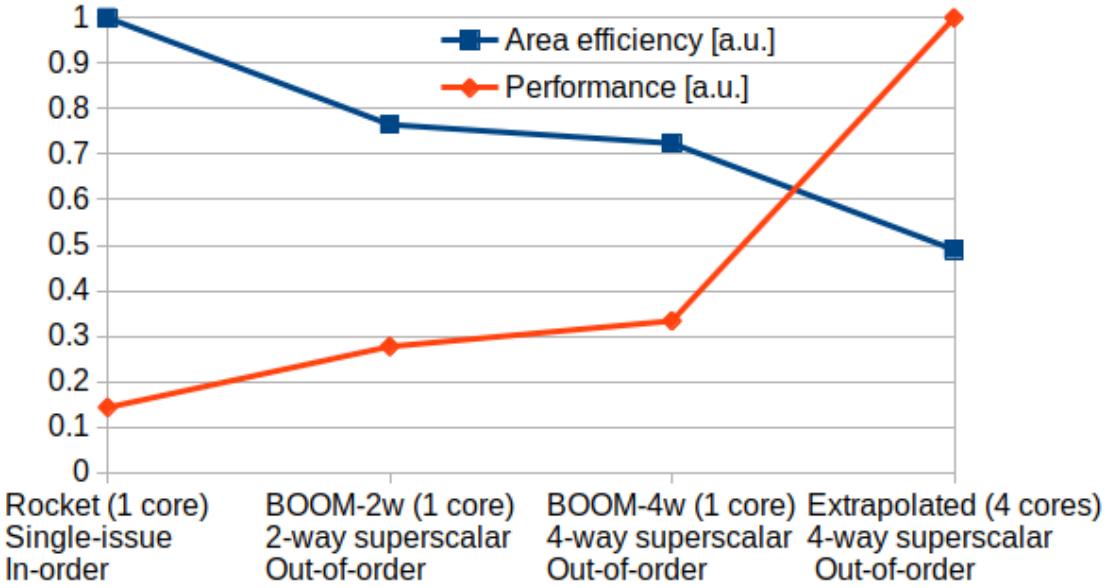
(Some of the) Standard Unprivileged Ext.	
M	Integer Multiplication and Division
A	Atomics
F	Single-Precision Floating-Point
D	Double-Precision Floating-Point
B	Bit Manipulation
C	16-bit Compressed Instructions
P	Packed-SIMD Extensions
V	General Vector Extensions

- E.g. RV64IMAFDC (aka RV64GC) with MSU for General Purpose processors
- RQ: How can we use the modularity of RISC-V in satellite data systems?
  - Which ISA **subsets**? And with what kind of **microarchitectures**?
    - Avoid non-standard custom extension: “A modified ISA is a new ISA”
  - Which level of performance? **And how to measure them?**

General Purpose (GP) Processors

Many processors are General Purpose: to run non-compute-intensive workloads on Unix-like OSs

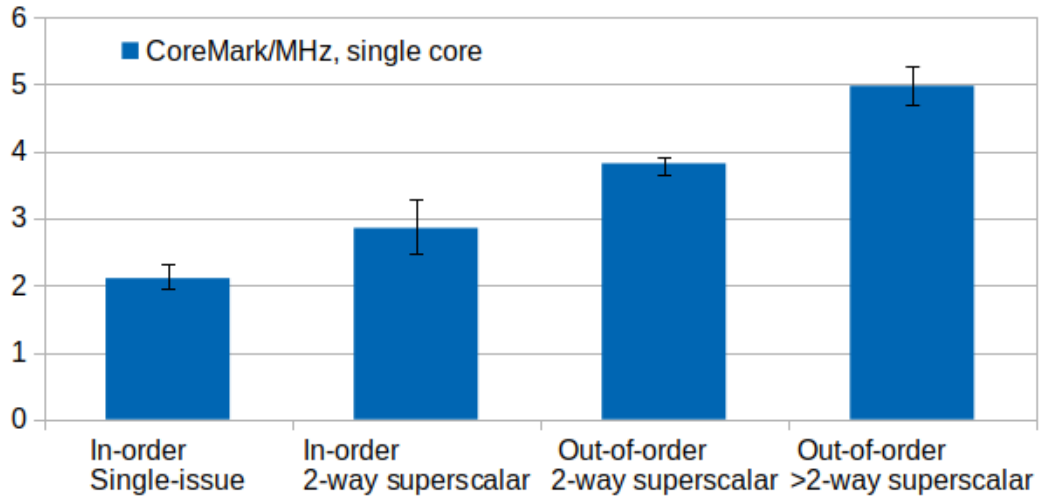
- HW support for virtual memory management
- Mainly best-effort basis
- Payload processors: many tasks, helping with the reuse of SW modules



- To increase performance:
  - **Pipelining**: increase max. frequency
  - **Instruction-level parallelism** (ILP): more instructions simultaneously
  - **Speculation**: i.e. assume an outcome and continue the execution instead of waiting
  - Use more complex **scheduling** of instructions: from in order to out of order
  - **Processor-Level Parallelism** (PLP), i.e. going multicore with Symmetric MultiProcessing
- The efficacy eventually **saturates**: find the right trade-off for the **target performance** and **acceptable area (and power) efficiency**.

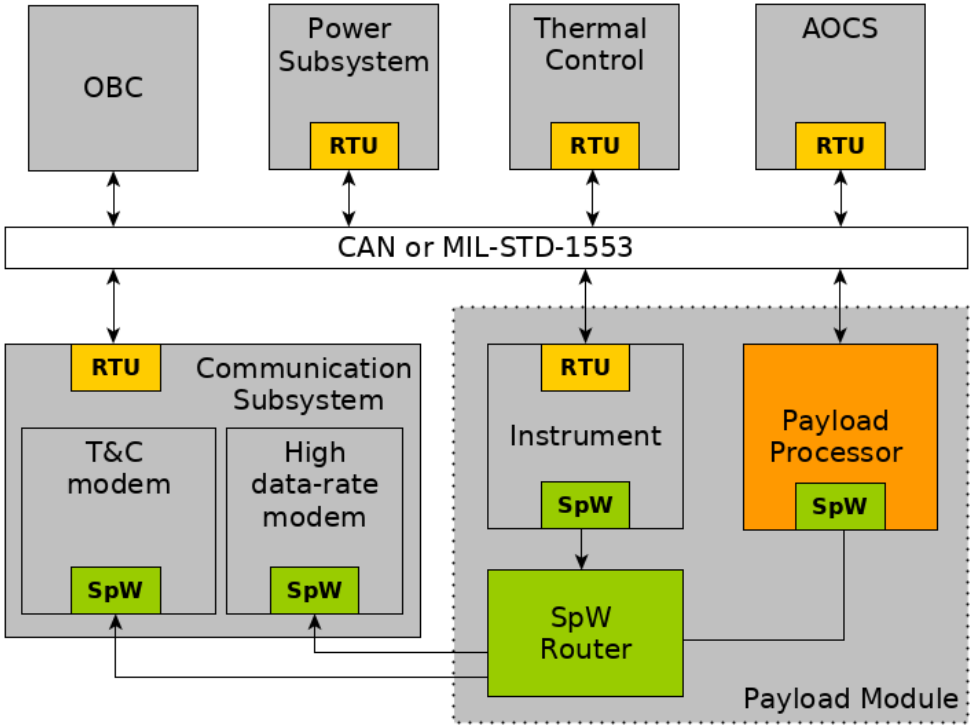
RISC-V profiles: GP processors

- Identify the features that impact the most on **performance** and **power/area efficiency**
- Define **profiles** for different target performance (**technology independence**)
  - Some extrapolations (e.g. doesn't measure the increase in maximum frequency, etc..).



Profile	Ref. Implement.	ISA subset (opt.)	Microarch.	PLP	Benchmark	Target Perf.
GP-LE	Rocket, ARM1176JZF-S LEON3/4	RV64GC	SI, IO	1-4	CoreMark	2-6 CM/MHz
GP-ME	ARM Cortex-A7 ARM Cortex-A8	RV64GC(P)	2-w, IO	1-4	CoreMark	3-9 CM/MHz
GP-HE	BOOM (2-w) ARM Cortex-A9	RV64GC	2-w, OoO	1-4	CoreMark	4-12 CM/MHz
GP-VHE	BOOM (4-w) ARM Cortex-A15	RV64GC(P)	4-w, OoO	1-4	CoreMark	5-15 CM/MHz

RISC-V Microcontrollers



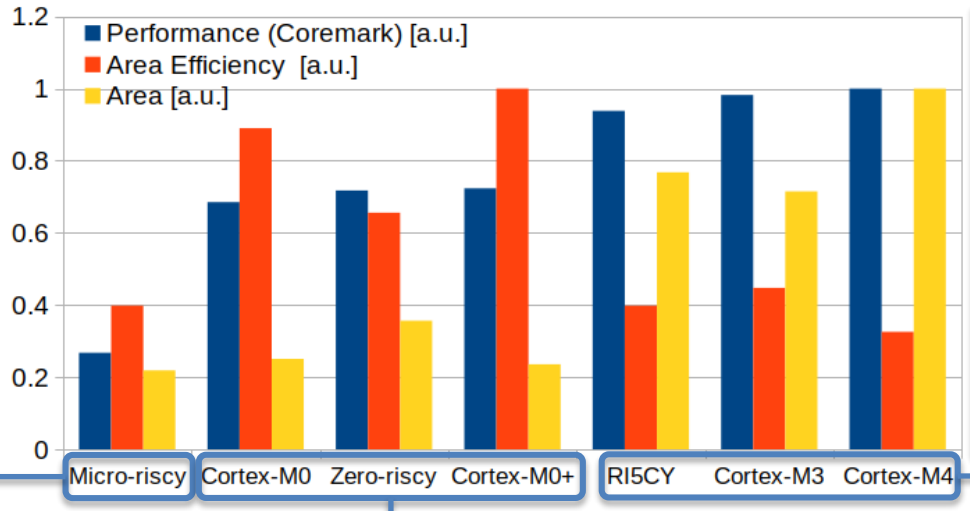
While a single-core RV64GC (GP-LE-1) requires 185 kGE, there are RV32EC implementations that require only around 10 kGE

Remote Terminal Units are often present to implement control loops locally

- Usually GP processors are deemed too large, power hungry and not time-deterministic enough
- RISC-V allows
  - Use of subsets (RV32E) for low-area implementations
    - -25% compared to RV32I
    - -40% if we remove also M
  - Compressed instructions (C) to reduce memory requirements
    - -30% code size (SPEC 2006)
  - Bit manipulation (B): reduces code size and speeds up control operations

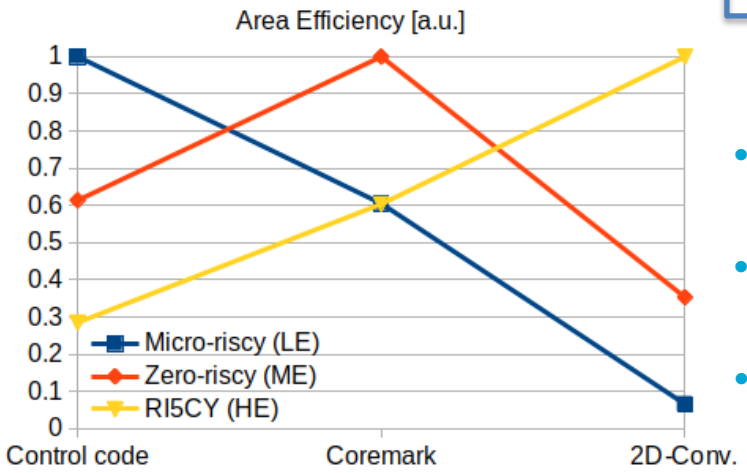
Benchmarking RISC-V microcontrollers

Low-End impl. (LE) for "pure control" applications (e.g. latchup protection of COTS)



High End impl. (HE) employ (non-standard in the case of RI5CY) Single Instruction Multiple Data (SIMD) instructions to accelerate compute-intensive workloads

Mid End impl. (ME): control applications requiring a bit more calculations (e.g. AOCS)



- ME are the most area-efficient for CoreMark.
- LE are the most area-efficient for "pure control" applications
- HE are the most area-efficient for integer convolutions.
  - Usign CoreMark to benchmark a processor for 2D-convolutions leads to a suboptimal choice

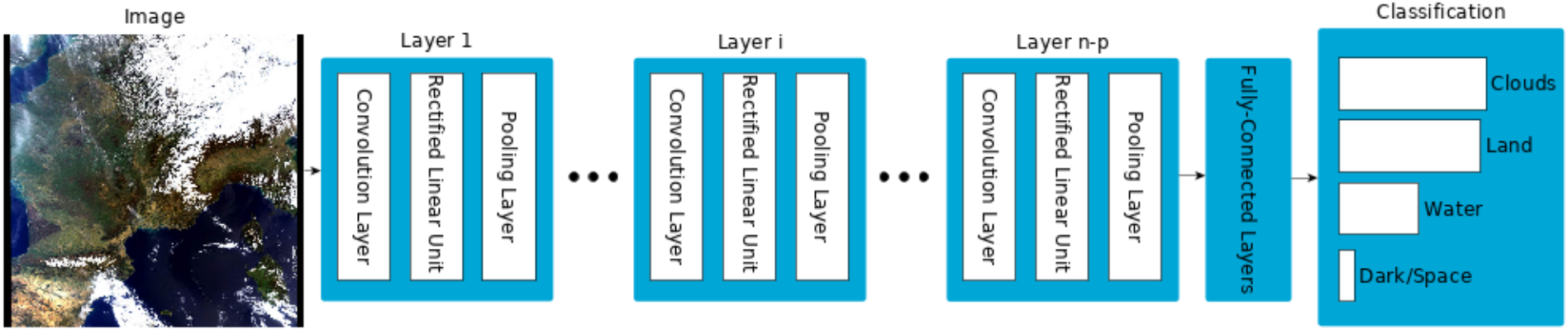
TU Delft

On-Board Decision-Making

- Do we need compute-intensive workloads on-board?
  - On-board encryption (security)
  - On-board data compression
  - On-board Digital Signal Processing (could be done also on ground)
  - **On-board decision-making**
- Simplest application for On-board Decision-Making: **data reduction**
  - Decision to be taken: **whether to send or not a certain image to Earth**
    - Downlink is a typical **bottleneck** of space data systems
    - Hours to days to downlink an image from a CubeSat to the ground
    - Storage space on CubeSats is typically very limited
- Many (creative) proposals from academia require on-board decision-making
  - Autonomous exploration (also overcomes latency of remote operations)
  - Active debris removal by vision-based navigation
  - Autonomous satellite swarms
  - Asteroid mining

From Images to Decisions

- In order to take decisions, images are ranked according to certain **features**, typically employing Convolutional Neural Networks (CNNs)
  - Composed by several layers that progressively abstract the data contained in the image

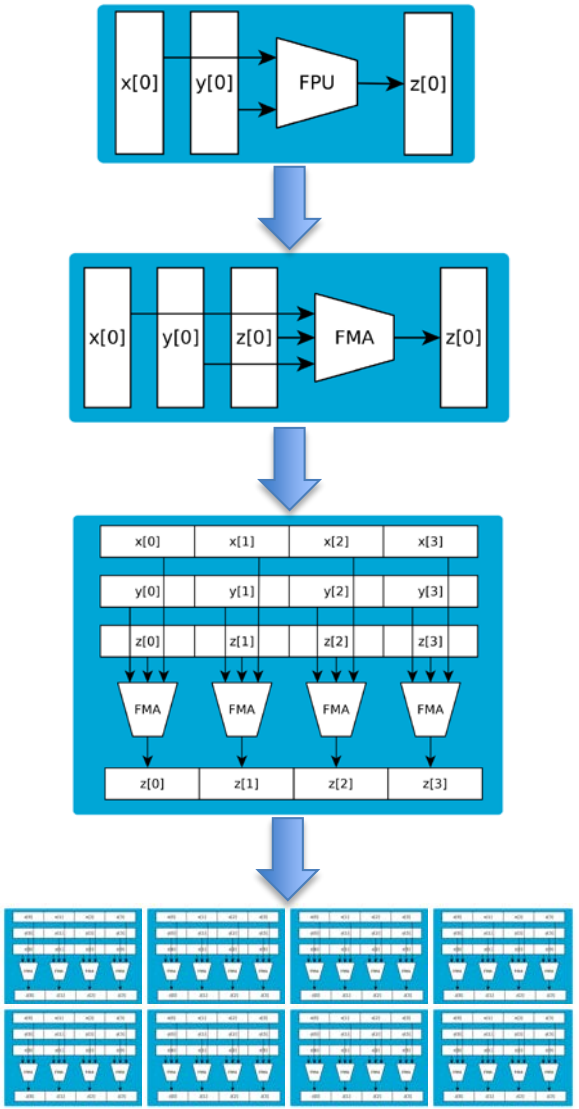


- Discrete convolutions are matrix operations and they are very **compute-intensive**
- Example for Layer 1:
  - Input image: 128x128 pixels (134x134 with padding) x 3 (R,G,B) Double-Precision (DP) elements
  - Kernel: 64 (filters) x 7x7x3 (coefficients in a filter) DP elements
  - Kernel size x 2 (Multiplication+Addition) operations per pixel: **338 MFLOP (DP)**
  - Read image (421 KiB) and filters coefficients (73.5 KiB), write result (8.8 MiB): memory traffic is **9.2 MiB**
  - Operational intensity (OI): **35 DP-FLOP/B**

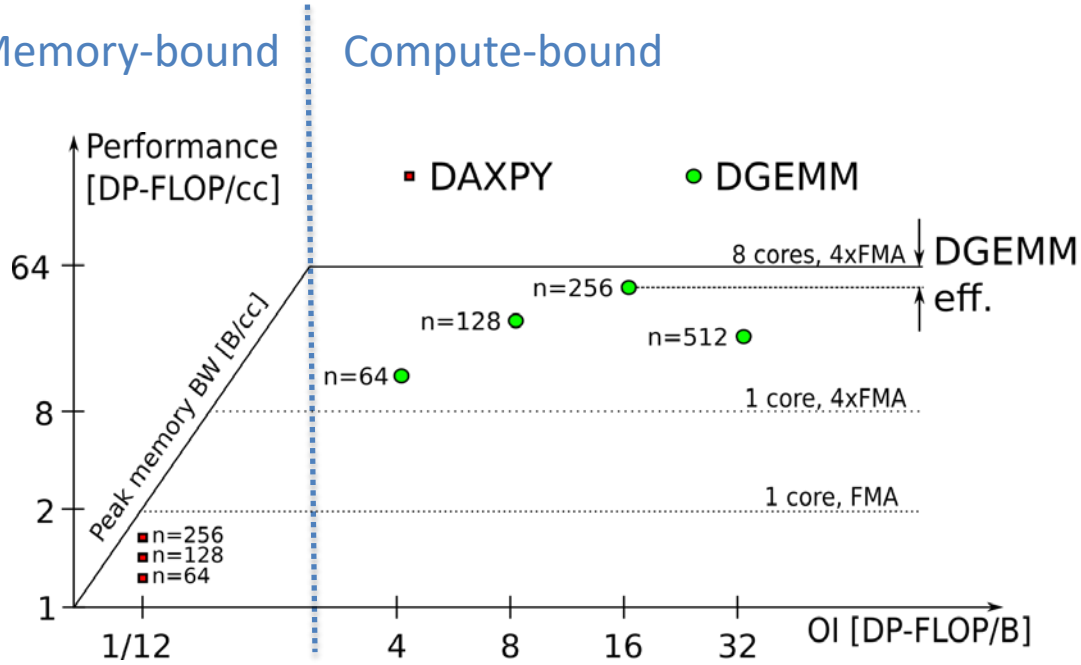
Increasing performance for compute-intensive workloads

Runtime determined by the number of **FLOPS**

- **Maximum Theoretical Perfomance (MTP)** in terms of FLOP/cc
  - **Independent** from ILP, speculation, caching, etc.
- To increase MTP, different approaches required:
  1. A single-core processor with a single FPU can achieve only up to **1 FLOP/cc**
  2. Introduce Fused Multiply-Add and Fused Multiply-Accumulate (**FMA**) instructions (e.g. $z \leftarrow x * y + z$), **MTP → 2 FLOP/cc**
  3. Replicate registers and FMA units to increase the Data-Level Parallelism (**DLP**)
     - With 4 FMAs, **MTP → 8 FLOP/cc**
  4. Replicate processing cores
     - With PLP=8, then **MTP → 64 FLOP/cc**



Roofline Model



- **Horizontal** line: MTP
- **Diagonal** line: peak memory bandwidth (BW)
- Classifies applications in memory or compute-bound
  - Compute-bound can be sped up with more DLP and PLP
- (Some) Benchmarks:
  - DAXPY: $y \leftarrow \alpha x + y$ (x and y vectors of length n, α scalar)
    - OI=1/12 DP-FLOPS, heavily **memory-bound**
  - DGEMM: $C \leftarrow \alpha A \times B + \beta C$ (A, B and C n×n matrices; α and β scalars)
    - OI proportional to n, for large n heavily **compute-bound**
- Performance are not ideal because the actual memory BW depends on:
  - Can the processor **issue instructions** fast enough to keep the FMAs busy all the time?
  - **Memory hierarchy**: does the data fit in the L1 cache? In the L2? Reads from DRAM?

RISC-V profiles: from microcontrollers to manycore processors

Profile	Ref. Impl.	ISA subset (opt)	Microarc.	PLP	Benchmark	Target Perf.
uC-LE	Zero-riscy	RV32E(M)C	2-3 st., SI, IO	1	Control Code	1 CM/MHz
uC-ME	Micro-riscy Cortex-M0(+)	RV32IMC	2-3 st., SI, IO	1	Coremark	2.4 CM/MHz
uC-HE (A-SC-LE)	RI5CY Cortex-M3/M4	RV32IMCP(F)	3-5 st., SI, IO	1	Int. Kernels	1 OP/cc
A-MC-LE	PULP	RV32IMCP(F)	3-5 st., SI, IO	8-32	Int. Kernels	4–16 OP/cc
A-MC-ME	Epiphany-IV	RV32IMCP(F)	3-5 st., SI, IO	64-256	Int. Kernels	32-128 OP/cc
A-MC-HE	Epiphany-V	RV32IMCP(F)	3-5 st., SI, IO	512-1024	Int. Kernels	256-512 OP/cc

Packed-SIMD proposal for floating point operations dropped, only for fixed point

Is there anything better than SIMD?

- Most DLP solutions on the market are SIMD
  - Easy to implement in HW
  - Encode DLP in the instruction
    - Portability is affected
  - High 'bookkeeping' overhead
- Vector processors are the opposite:
  - Code runs always at the max. DLP possible
  - Hardware is more complex
  - Proven more efficient in late 70's supercomputers
  - CMOS scaling made simpler solutions preferable
- Renovated interest with the end of Moore's Law
  - ARM released a vector ext. for ARMv8-A
    - Next Fujitsu's supercomputer
  - RISC-V Vector ext. (V) is one of the most anticipated extensions

DAXPY: $y \leftarrow \alpha x + y$

Packed-SIMD

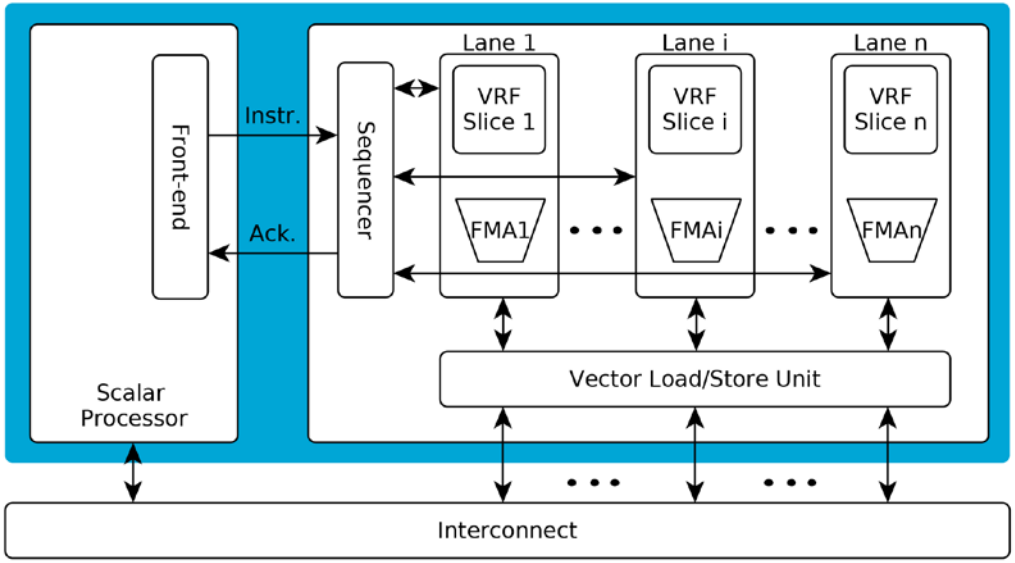
1: Load scalar
2: Determine maximum length usable (SW)
3: Jump to routine with the length found
…
i+1: Replicate scalar in SIMD registers
i+2: Load (part of) vector x
i+3: Load (part of) vector y
i+4: Length-dependent array FMA instr.
i+5: Store result
i+6: Decrease/increase counters
i+7: If not done, jump to row 2
…

Vector processor

1: Load scalar
2: Set maximum vector length possible (HW)
3: Load (part of) vector x
4: Load (part of) vector y
4: Length-agnostic scalar-vector FMA instr.
5: Store result
6: Decrease/increase counters
7: If not done, jump to row 2

RISC-V Vector Processors

- The RISC-V V ext. adds:
  - Vector Register File (32x64xlanes b):
    - Configurable number of element of configurable length
  - Integer, Floating and Fixed-Point operations (also FMA)
    - 3 vectors, 2 vectors and a scalar, etc.
    - Other instructions to deal with vectors
- Typically implemented replicating identical lanes
  - E.g. 1 FMA and 1 ALU per lane
  - Operate in lockstep (simple control)
  - Good scalability (at least up to 16 lanes)



Platform	GP+FMA (PLP=4)	A-M-LE (PLP=8)	GPU Mobile	GPU Desktop	RVV SC (16 lanes)
MTP [DP-FLOP/cc]	8	64	365 (SP)	2k	32
DGEMM Eff.	75%	30-70%	56% (SGEMM)	60-80%	>90%

Conclusion

- RISC-V is an open and modular ISA with already a relatively large user base
  - In this presentation we explored its potentialities, from tiny microcontrollers to high-performance processors for AI
- Other research questions to be answered:
  - How to increase **performance**, **power and area efficiency** of RISC-V processors to meet stringent requirements in satellite data systems?
  - **Technology readiness**: How to bring novel high-performance architectures from terrestrial applications to (at least) radiation-tolerant components?
- More (and references) in: S. Di Mascio et. al, “Leveraging the Openness and Modularity of RISC-V in Space”, Journal of Aerospace Information Systems, Volume 16, Issue 11, 2019
- Thanks to Cobham Gaisler and ESA

Thank you for your attention!

Questions?