Advanced Dynamic Scalarisation for RISC-V GPGPUs

Matthew Naylor, Alexandre Joannou, Theo Markettos, Paul Metzger, Simon Moore, Tim Jones
University of Cambridge, UK

(ICCD 2024)

GPGPU research

Software simulators are often used to model proprietary hardware at the intermediate language level, making it **easy to overlook effects on**:

- microarchitecture
- synthesis quality
- machine code

Open-source GPGPU hardware

Implementation	Year	ISA		
MIAOW	2015	AMD	Proprietary ISA, preventing free deployment of hardware	1
FlexGrip	2013	Custom	No access to mature software stack or compiler	2
FGPU	2016	Custom		
NyuziRaster	2016	Custom		
Simty	2017	RISC-V	Free, unencumbered ISA with rich software stack & compiler	3
Vortex	2021	RISC-V		

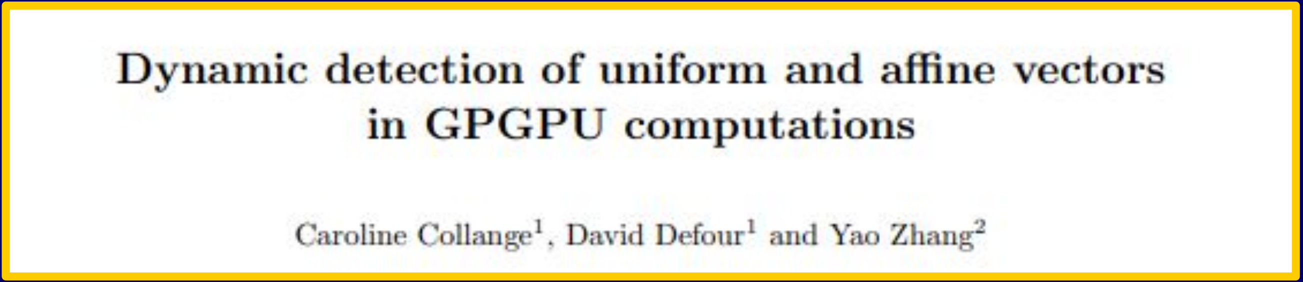
Single Instruction, Multiple Threads (SIMT)

Execute multiple hardware threads (a **warp**) in lockstep, exploiting:

- control-flow regularity
- memory-access regularity
- value regularity

Not yet exploited in RISC-V GPGPUs

What is value regularity?



Dynamic detection of uniform and affine vectors in GPGPU computations

Caroline Collange[1], David Defour[1] and Yao Zhang[2]

Seminal study from 2009 reports:

- 27% of register reads yield the same value for each thread in a warp (**uniform vectors**)
- 44% yield values separated by a constant stride (**affine vectors**)

Unoptimised, this leads to a large amount of wasted storage, computation, and energy.

Where does value regularity come from?

In CUDA, each thread typically:

- ➢ uses its **thread index** within a block
- ➢ and its **block index** within a grid

to determine which part of the input to read and which part of the output to write. For threads in a warp:

- ➢ the **thread index** is affine
- ➢ the **block index** is uniform
- ➢ and uniform/affine vectors often propagate

Contributions

A new open-source, synthesisable, RISC-V GPGPU that exploits value regularity through **advanced dynamic scalarisation**:

- **Register-file compression**
  (for reduced storage / silicon area)

- **Parallel scalar and vector pipelines**
  (doubling peak IPC with one extra execution lane)

No ISA or compiler extensions required!

SIMTight, our new RISC-V GPGPU design

Is an **RV32IMAxZfinx** streaming multiprocessor

- ➢ 64 warps and 32 threads per warp by default
- ➢ Warp scheduling
- ➢ Branch divergence and reconvergence
- ➢ Memory-access coalescing
- ➢ Scratch memory with parallel random access

And ships with a **CUDA-like programming API**

- ➢ And a suite of 14 benchmark programs

Register-file compression

SIMTight detects uniform and affine vectors in hardware and stores them compactly in a **scalar register file** (SRF).

- General vectors are allocated dynamically in a sized-constrained **vector register file** (VRF).
- If the VRF becomes full, vector registers are **automatically spilled** to main memory.

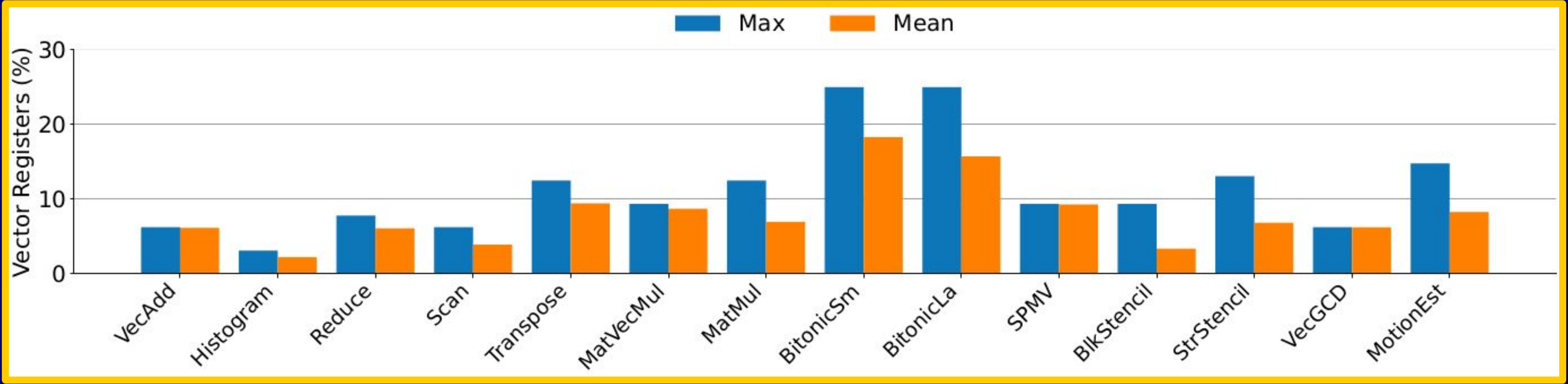
Parallel scalar and vector pipelines

SIMTight includes both a scalar pipeline and a general vector pipeline, operating independently and in parallel:

- Instructions that map affine operands to affine results are said to be **scalarisable**
- The scalar pipeline can execute scalarisable instructions using a **single affine execution lane**
- Warps are **automatically** moved between the pipelines using a **prediction table**.

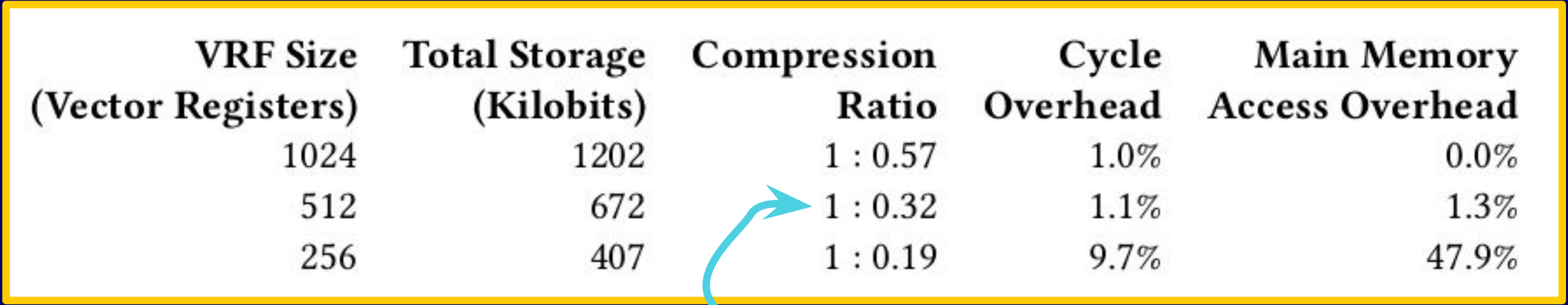
Results

Vector occupancy in the register file



Proportion of registers stored as vectors in the VRF (geomean 12%). Remaining registers are stored compactly as scalars in the SRF.

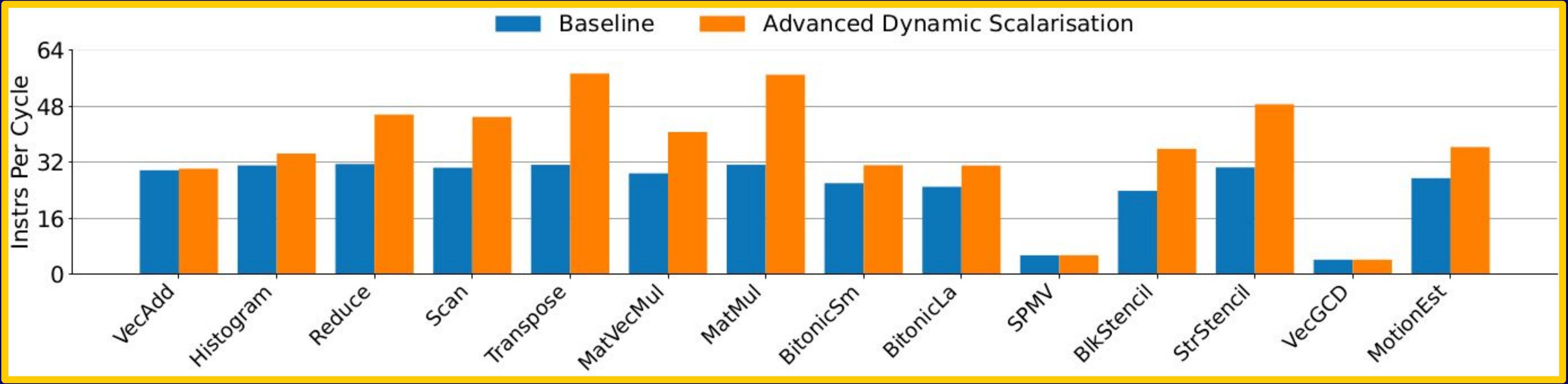
Register-file compression



VRF Size (Vector Registers)	Total Storage (Kilobits)	Compression Ratio	Cycle Overhead	Main Memory Access Overhead
1024	1202	1 : 0.57	1.0%	0.0%
512	672	1 : 0.32	1.1%	1.3%
256	407	1 : 0.19	9.7%	47.9%

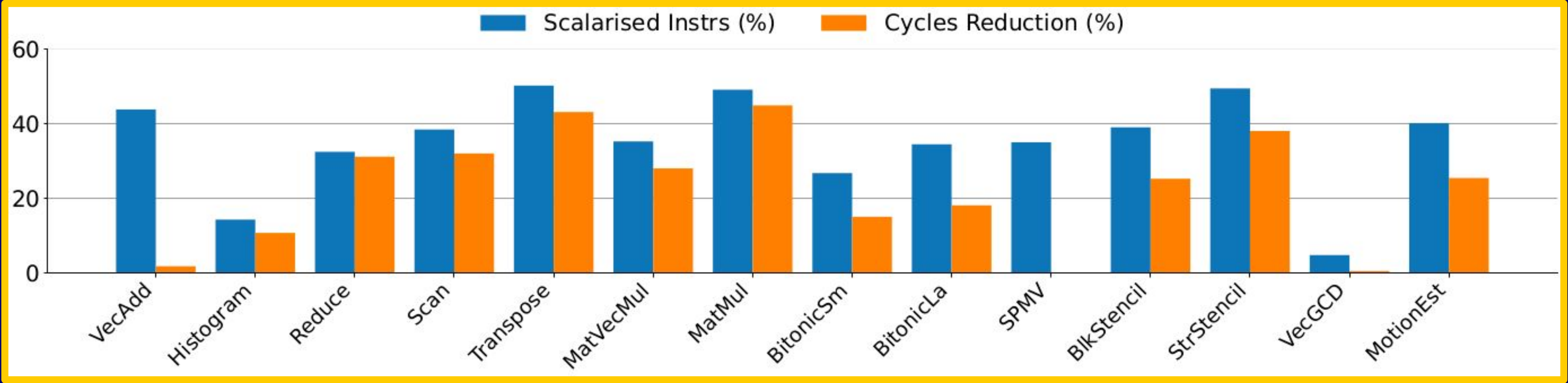
Reduces register file storage by 68% (178KB per SM) for a geomean 1% cycle overhead

Instruction throughput



Baseline IPC often approaches warp size (32), but surpasses it with parallel scalar and vector pipelines

Parallel scalar and vector pipelines



Parallel scalar and vector pipelines enable 31% of instructions to be scalarised for a 24% reduction in execution time (clock cycles)

Synthesis results (on FPGA)

Configuration	Use DSP Blocks?	Fmax (MHz)	Area (ALMs)	Area (DSPs)
Baseline - FP	Yes	207	48K	66
Baseline	Yes	204	94K	297
Baseline + RFC	Yes	196	100K	297
Baseline + RFC + PP	Yes	194	103K	299
Baseline - FP	No	205	69K	0
Baseline	No	205	176K	0
Baseline + RFC	No	191	181K	0
Baseline + RFC + PP	No	189	188K	0

5% Fmax overhead (wall clock reduction is 20% rather than 24%)

7% logic-area overhead

Conclusion

Emerging RISC-V GPGPUs can exploit value regularity effectively, at low hardware cost, without specialised ISA and compiler support

https://github.com/CTSRD-CHERI/SIMTight

Funding

UKRI Engineering and Physical Sciences Research Council

CAPcelerate (EP/V000381/1)

Chrompartments (EP/X015963/1)

Funded by the Digital Security by Design (DSbD) Programme delivered by UKRI to support the DSbD ecosystem.