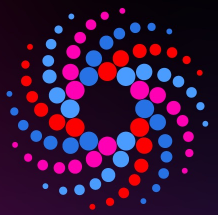


SC22
Dallas, TX | hpc accelerates.

An Initial Evaluation of Arm's Scalable Matrix Extension

Finn Wilkinson & Prof. Simon McIntosh-Smith

University of Bristol

(fw17231@bristol.ac.uk, s.mcintosh-smith@bristol.ac.uk)

Background

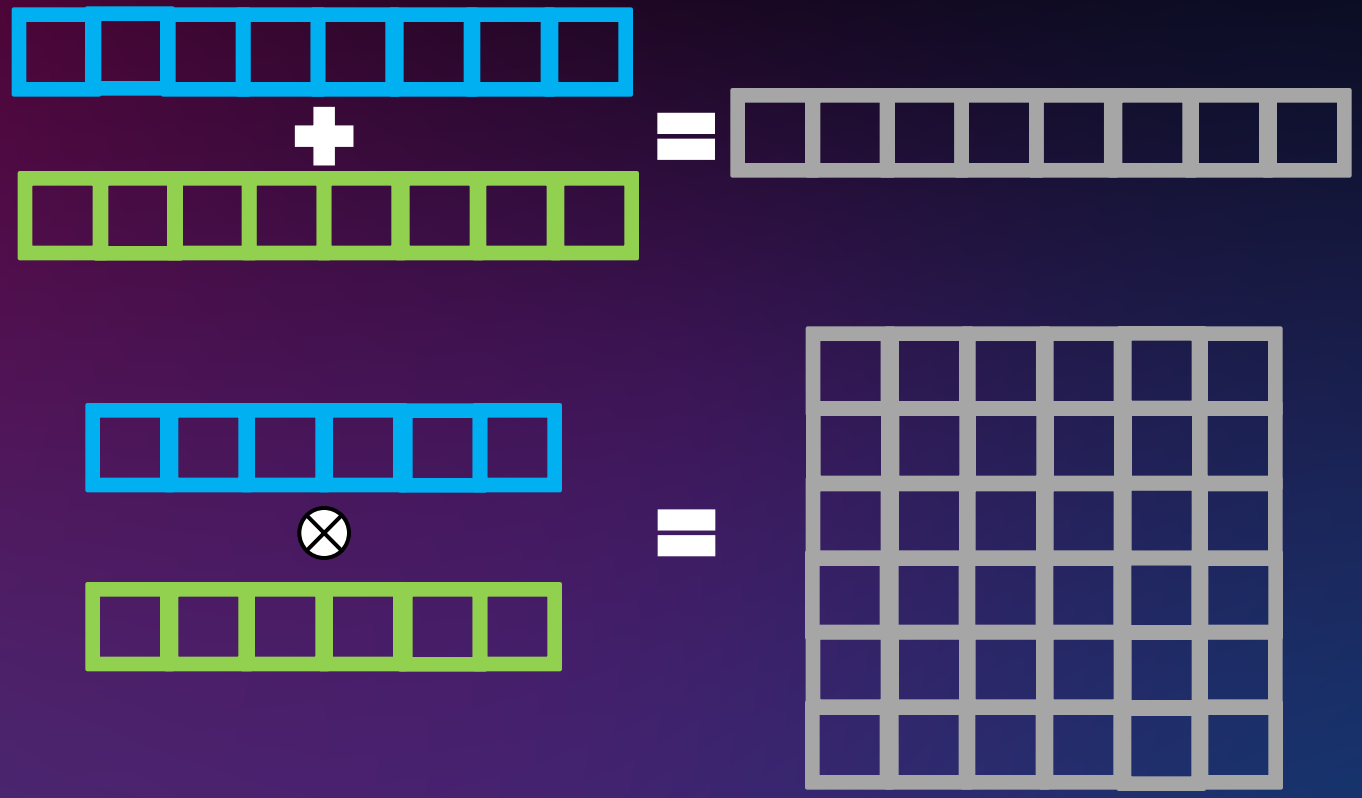
- BLAS operations form a large part of many HPC applications
  - Whether this is Level 1 vector-vector operations, or Level 3 matrix-matrix operations
- Whilst GPUs & DSAs tend to provide the best performance for these workloads, only 34% of systems[1] use any form of accelerator
  - Typically requires additional work from the programmer to target an accelerator
  - An additional memory transfer overhead from CPU -> accelerator
- Developing new in-CPU BLAS acceleration techniques is still a relevant area to pursue
  - Additional performance gains without extra work for programmers
  - Additional performance gains without additional hardware

University of BRISTOL

[1] - June 2022 Top500

Background – CPU BLAS Acceleration

- SIMD and Vector Extensions :
  - Arm's NEON and SVE extensions
  - X86's AVX-512
  - RISC-V's RVV
- Matrix Extensions :
  - Apple's AMX
  - Intel's AMX-512
  - IBM Power10's MMA
  - Arm's SME



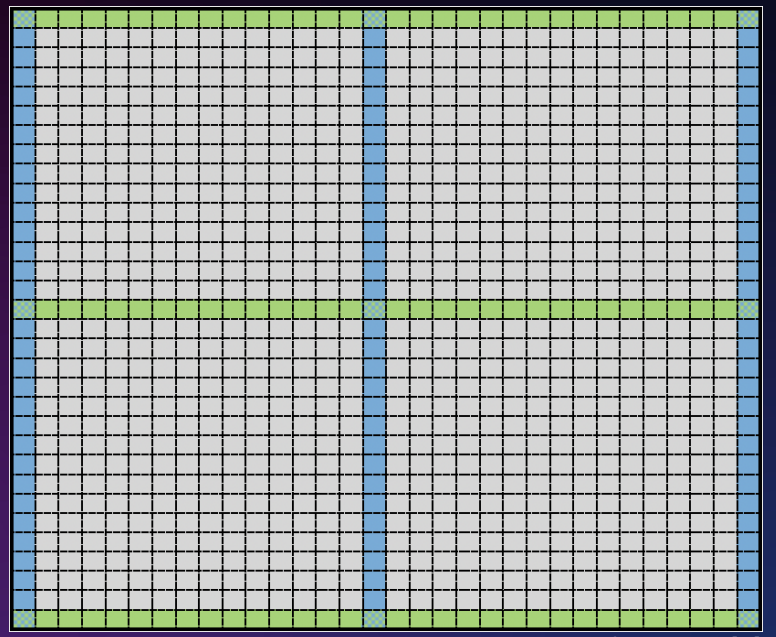
Background - Arm's Scalable Matrix Extension

- Builds upon Arm's SVE
  - Utilises SVE *Z* registers and *P* registers as source operands
- A single matrix register is introduced - ZA
- Introduces two new processing modes :
  - `PSTATE.SM` - SVE Streaming-Mode; provides a context switch to enable SME instructions
  - `PSTATE.ZA` - Enables access to the SME matrix register ZA
- SME has its own vector length – the *Streaming Vector Length (SVL)*
  - When `PSTATE.SM` is enabled, SVE instructions will use SVL rather than VL
  - SVL = {128, 256, 512, 1024, 2048}-bits

University of BRISTOL

Background – SME's Matrix Register

- ZA is a 2D matrix with dimensions [SVL_B x SVL_B]
  - SVL_B = number of bytes in SVL
- Number of available rows fixed to SVL_B
- Number of available columns (elements-per-row) can change with the size of each element
  - 1 byte per element = SVL_B columns
  - 4 bytes per element = SVL_B/4 columns (i.e. FP32)
- Accessed via individual rows or columns, or via tiles
  - A tile is a Square 2D sub-array of ZA
  - The number of tiles available is dictated by the element size



The ZA register - Image from Arm [2]

University of BRISTOL

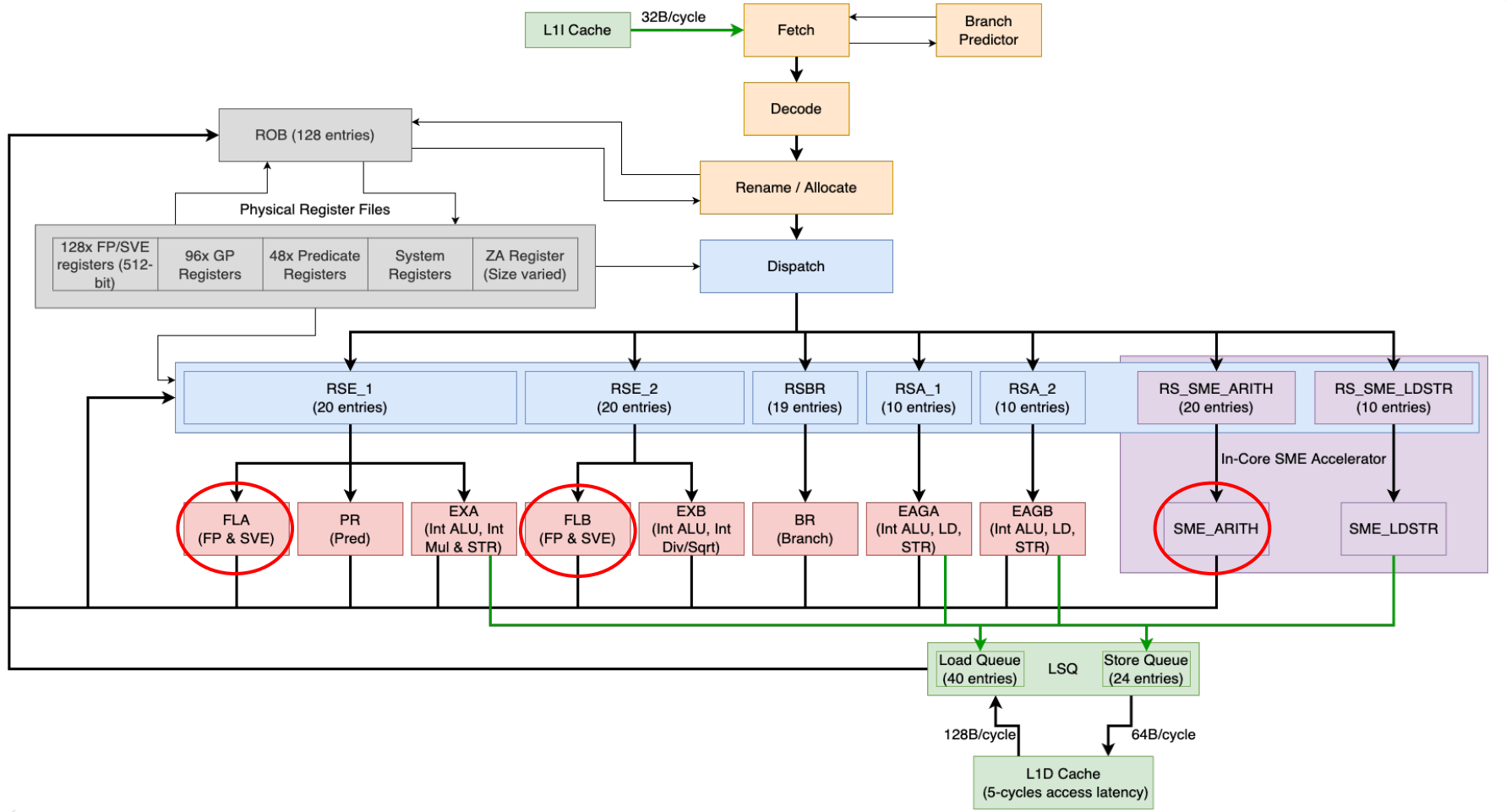
[2] - https://developer.arm.com/documentation/ddi0616/latest

Evaluation Methodology – Simulation Environment

- Used the **Simulation Engine (SimEng)** cycle-level architectural simulator
  - Easy to modify & configure core models
  - Fast simulation speed (~1MIPS)
  - Accurate to read-world scenarios
- Implemented support for SME (+ SVE2) within SimEng via Capstone-Engine update
  - SME support within SimEng to be included in next release (before end of year)
- Used a core model of Fujitsu's A64FX as the basis of a hypothetical core with SME support
  - Native SVE support removes some guess work with SME configuration details

https://github.com/UoB-HPC/SimEng

Evaluation Methodology – Hypothetical Core Model



- Infinite L1$ modelled as this study focuses on compute only
- A64FX's SVE single core throughput = 64 FP32 FLOPs/cycle (FMLA)
  - Has 2x512-bit SVE EU's
- In-core SME Accelerator's throughput = 512 FP32 FLOPs/cycle (FMOPA)
  - While SVL = 512-bit

Evaluation Methodology – Target Workload

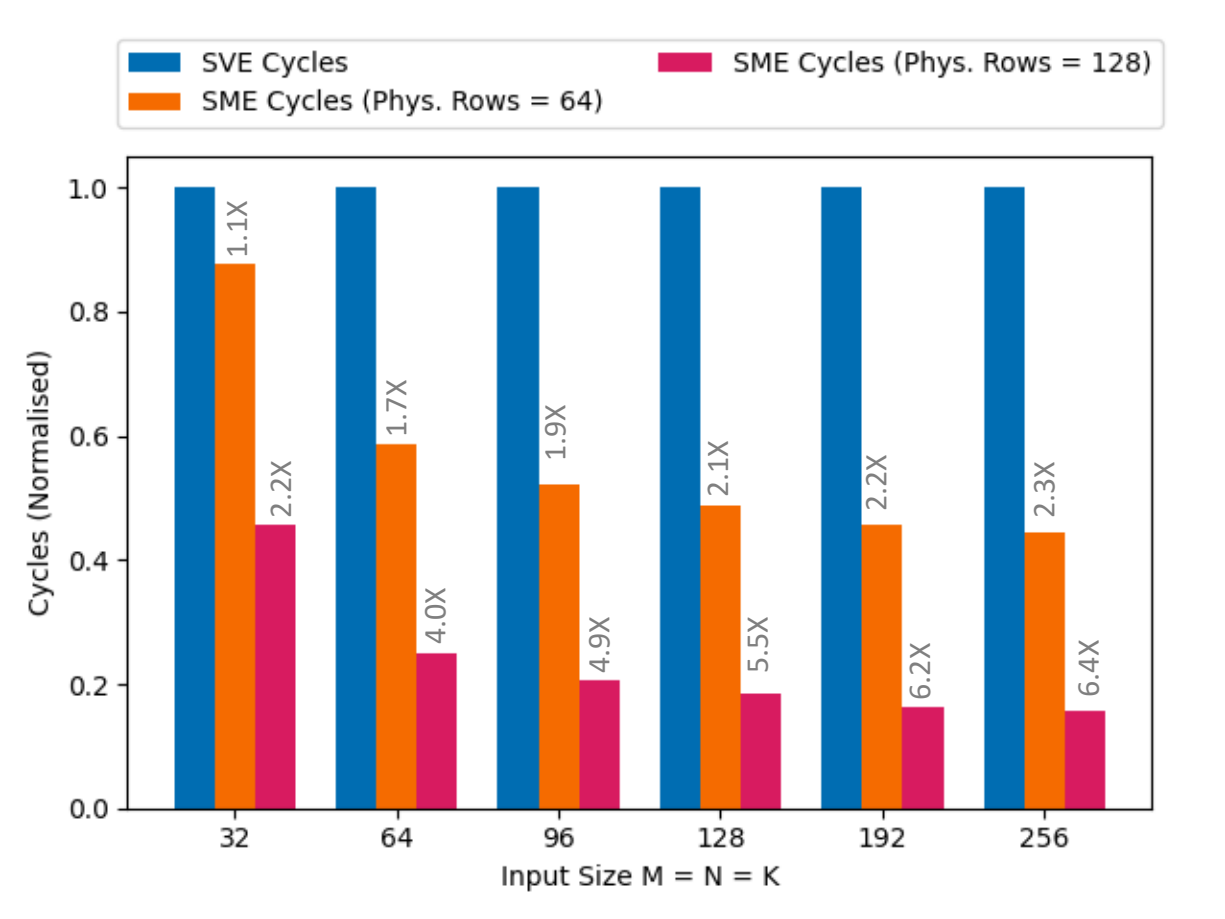
- An optimized FP32 Matrix Multiplication kernel C=AB
  - Lack of compiler & library support for SME limits workloads
- Two versions of the workload; one utilising SVE, another utilising SME
  - SVE kernel is adapted from Arm's SVE documentation, SME kernel is adapted from assembly provided by Arm
  - Both MatMul function versions self verify against a naïve MatMul implementation
- The SME version involves a pre-processing step to transpose input matrix A
  - SME MatMul is calculated as sum of outer products, transposing allows for row-major access of A and B
- $$C = AB = \sum_{i=0}^{K} \boldsymbol{a}_i^T \boldsymbol{b}_i$$

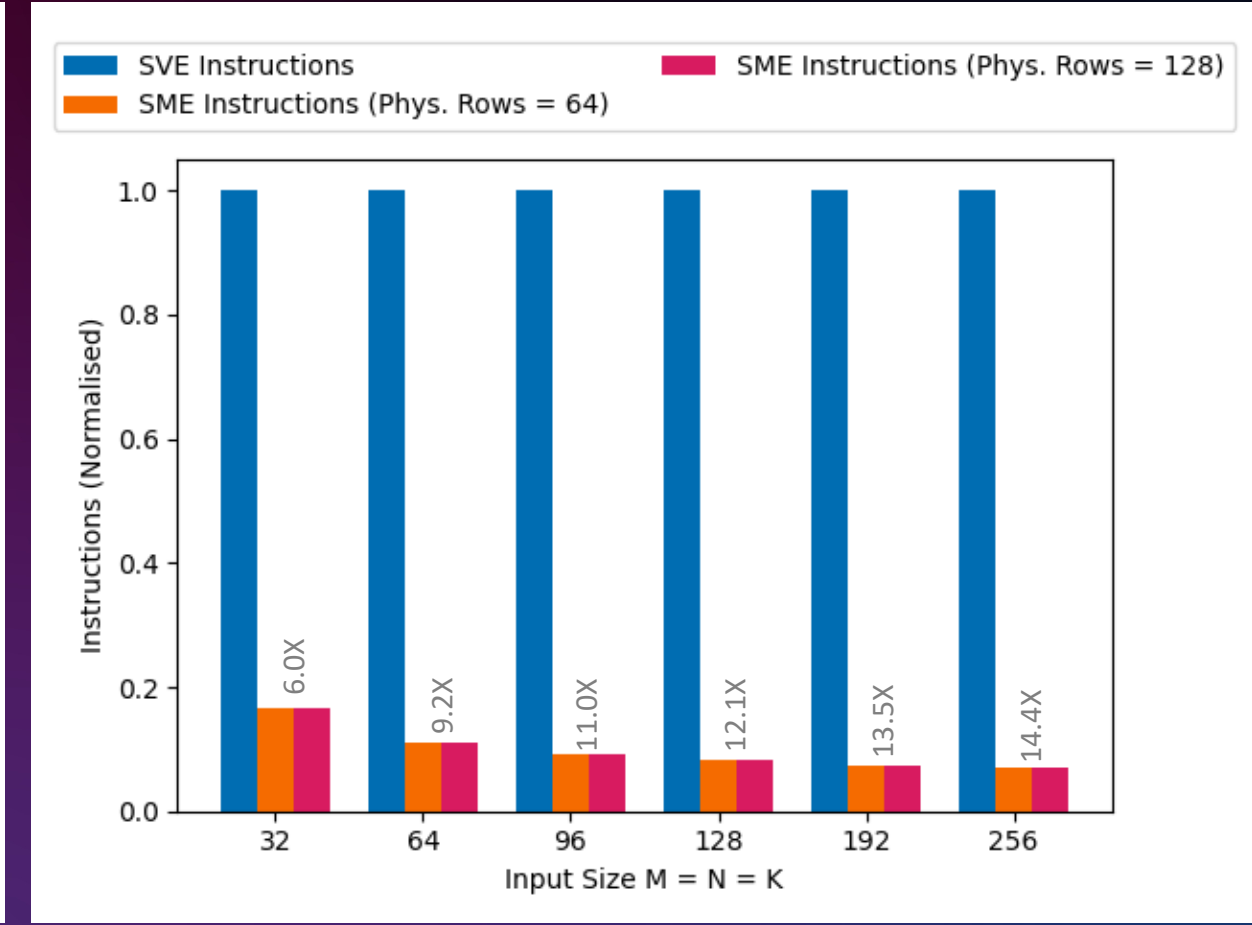
Where A, B, C are matrices of dimensions (M x K), (K x N), (M x N) respectively;
$\mathbf{a}_i$ is column *i* of A, $\mathbf{b}_i$ is row *i* of B

University of BRISTOL

Results – SVE vs. SME

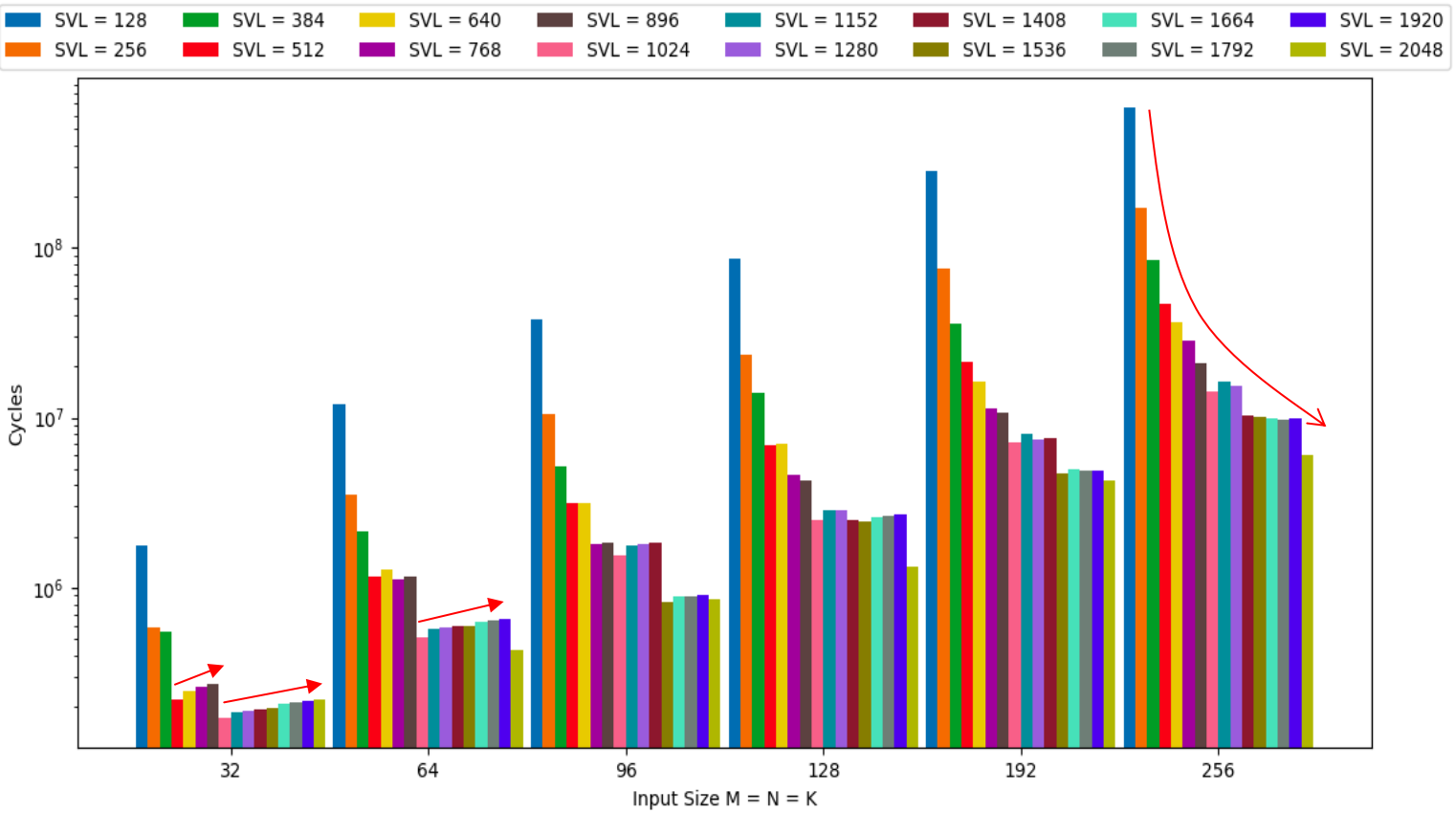
VL = SVL = 512-bit; For each input, the result is calculated 200 times per run. Lower is better.





Results – Comparing SVL Values

Physical Reg. Count = 2xSVL_B ; For each input, the result is calculated 200 times per run. Lower is better.



- As SVL increases, perf. gain plateaus
- Step-like drops due to SVL not being divisor of input size
- Gradual increase in cycles between some SVL values is due to increasing amount of redundant instructions as SVL grows

University of BRISTOL

Summary

- We have provided first-of-a-kind performance results for Arm's Scalable Matrix Extension
- For dense MatMul, our hypothetical core saw up to a **6.4X*** speedup when utilising SME
  - Perhaps somewhat unsurprising…
  - Provides a good starting point for further evaluation
- A greater SVL does not guarantee a performance improvement, especially for smaller input sizes
  - Our SME MatMul implementation is susceptible to performing redundant work

University of BRISTOL

*Max speedup quoted in paper is 9.8x due to a bug in the simulator discovered after submission.

Thank you for listening!

Finn Wilkinson

University of Bristol HPC Group

fw17231@bristol.ac.uk

https://github.com/UoB-HPC/SimEng

University of BRISTOL