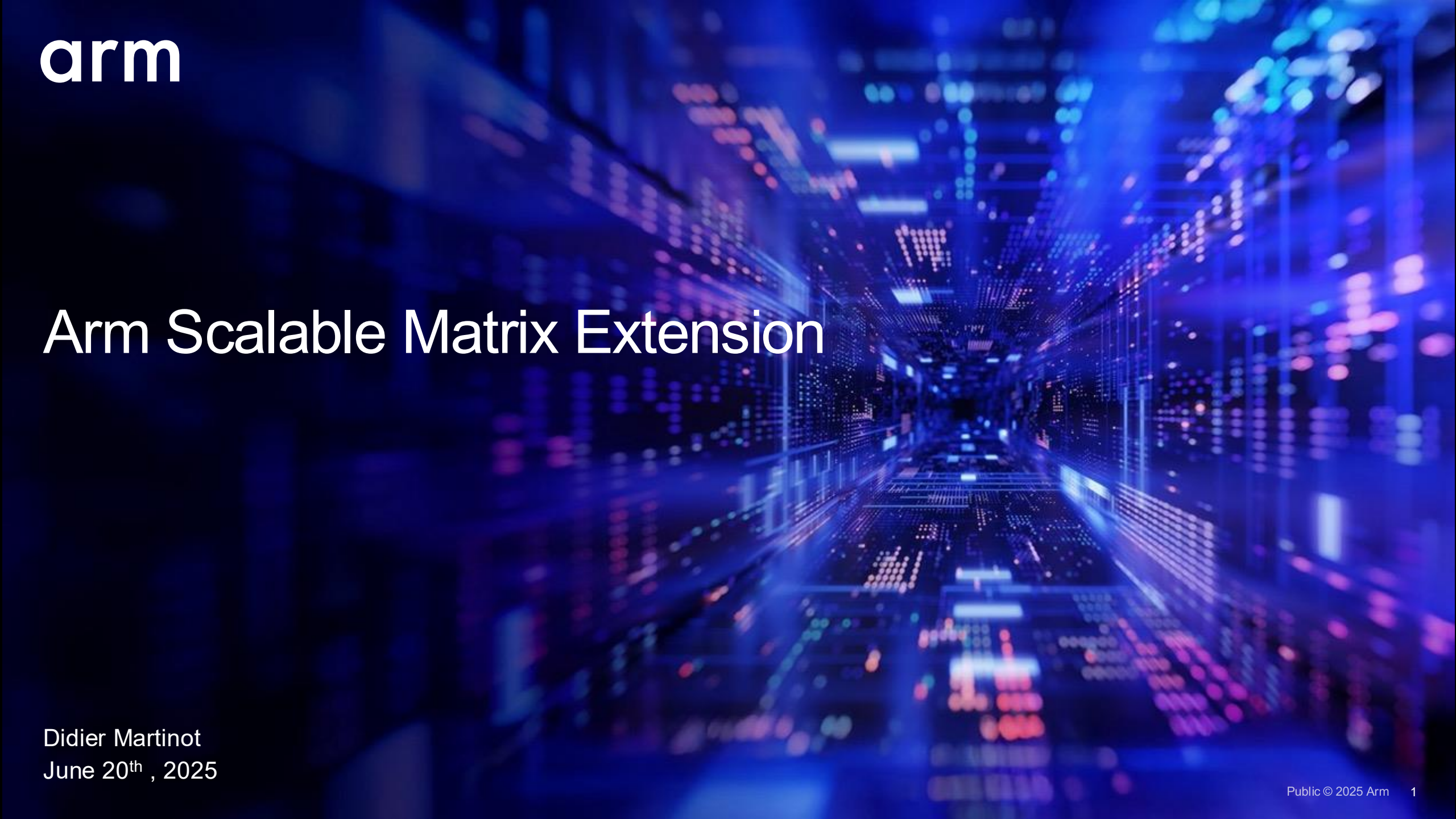


arm

Arm Scalable Matrix Extension

Didier Martinot

June 20th , 2025

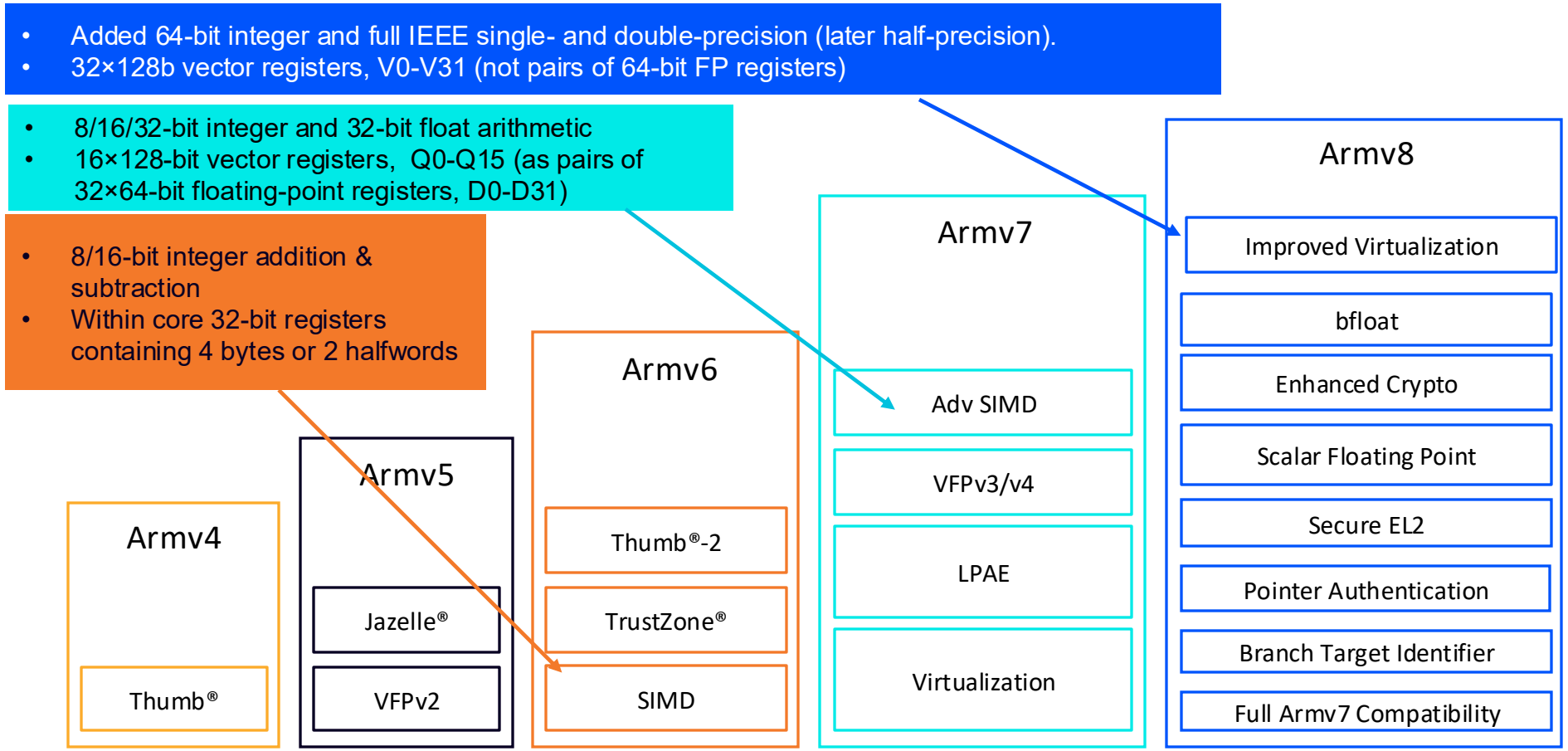
Table of Content:

- High level introduction of Scalable Matrix Extension (SME)
- Presentation of some use-cases leveraging SME technology

Scalable Matrix Introduction – High-Level Introduction

The Arm Architecture: Continually Innovating and Evolving Vector Processing -

- Quick Recap on the early stage of the journey…



From Vector Processing to Matrix Processing

As of now, NEON is essential for many key workloads running in a CPU:

Scientific simulations

Machine Learning

Digital Signal Processing

Computer Vision

Augmented and Virtual Reality

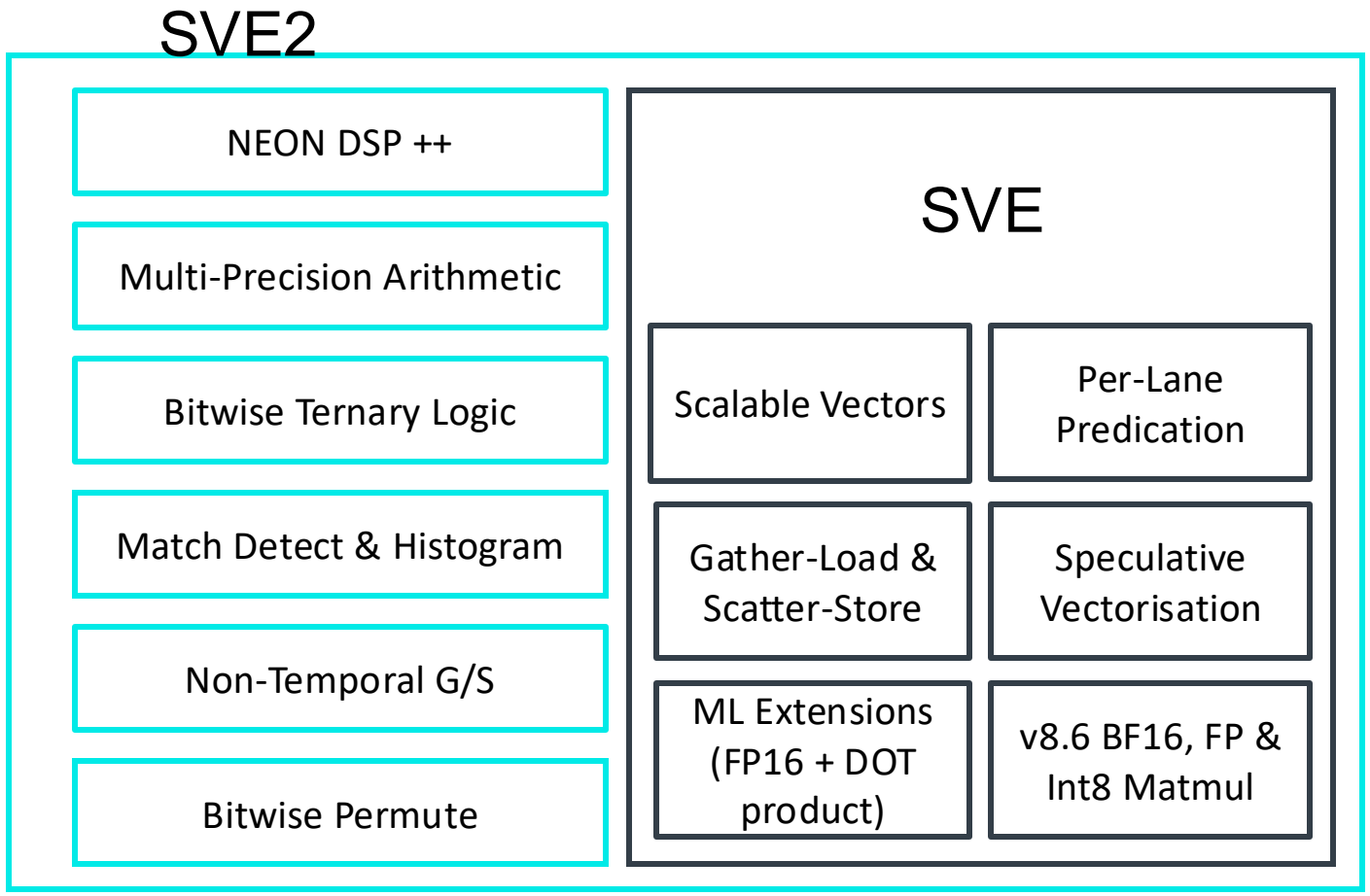
Yet - a lot of these applications require more and more vector processing !

SVE and SVE2(Armv9)

- NEON cannot be extented to higher vector length and there is a limit to the amount of instruction parallelism you can implement in a CPU (at reasonable cost)
  - Need of SIMD capability the VL of which can be decided by the u-architecture beyond 128.

- Armv8 - Scalable Vector Extension (SVE) for the High Performance Computing market
  - Enables scientific workloads such as climate modelling, vaccine research and materials research
  - Enables Vector Length Agnostic (VLA) programming
  - Tackles traditional barriers to auto-vectorization

- Armv9 SVE2 extends the functionality to cover all traditional NEON use cases



From Vector Processing to Matrix Processing

Vector Extension Architecture is now future proof to serve many key workloads running on CPU:

Scientific simulations

Machine Learning

Digital Signal Processing

Computer Vision

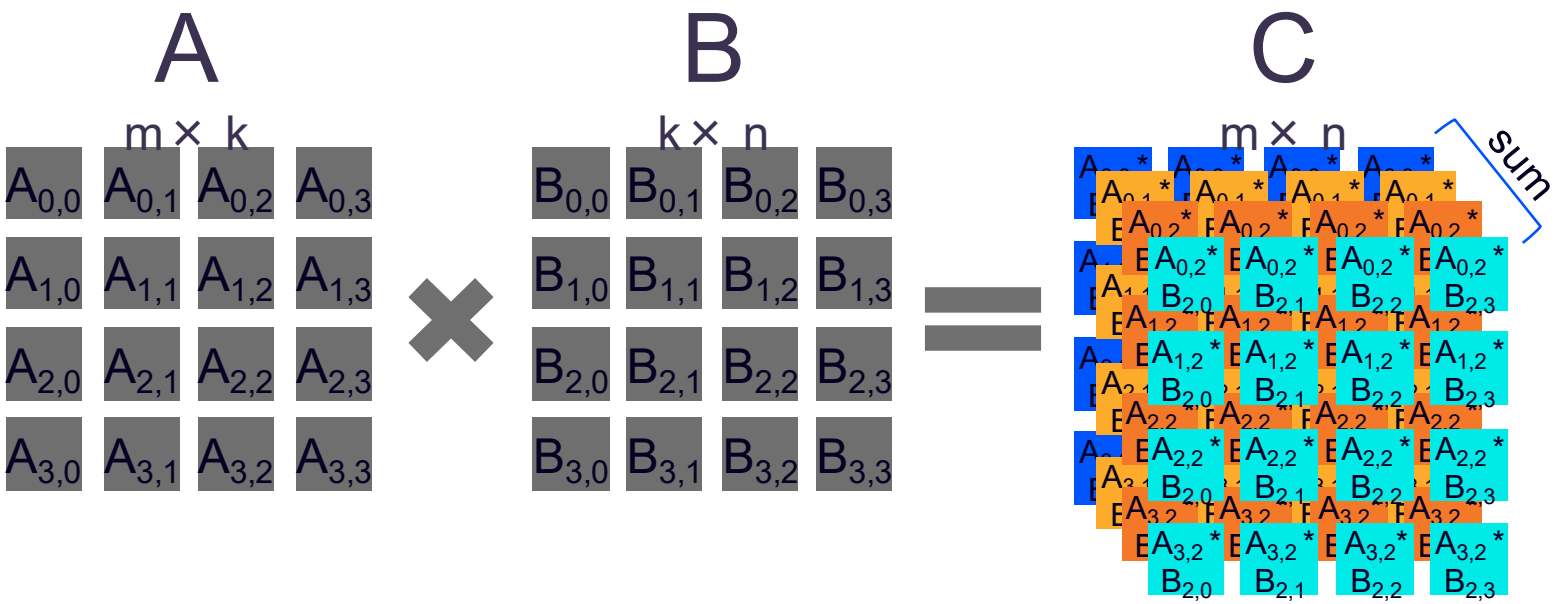
Augmented and Virtual Reality

Yet - a lot of these applications are working not only on vectors but also on matrices and multidimensional tensors !

Matrix Multiplication as the Sum of Vector Outer Product

- We typically learn to perform matrix multiplication by performing independent dot products (one result at a time)
- But, a matrix product can be expressed as a sum of vector outer products:

$$AB = \sum_{k=1}^{p} a_k^{col} \otimes b_k^{row}$$

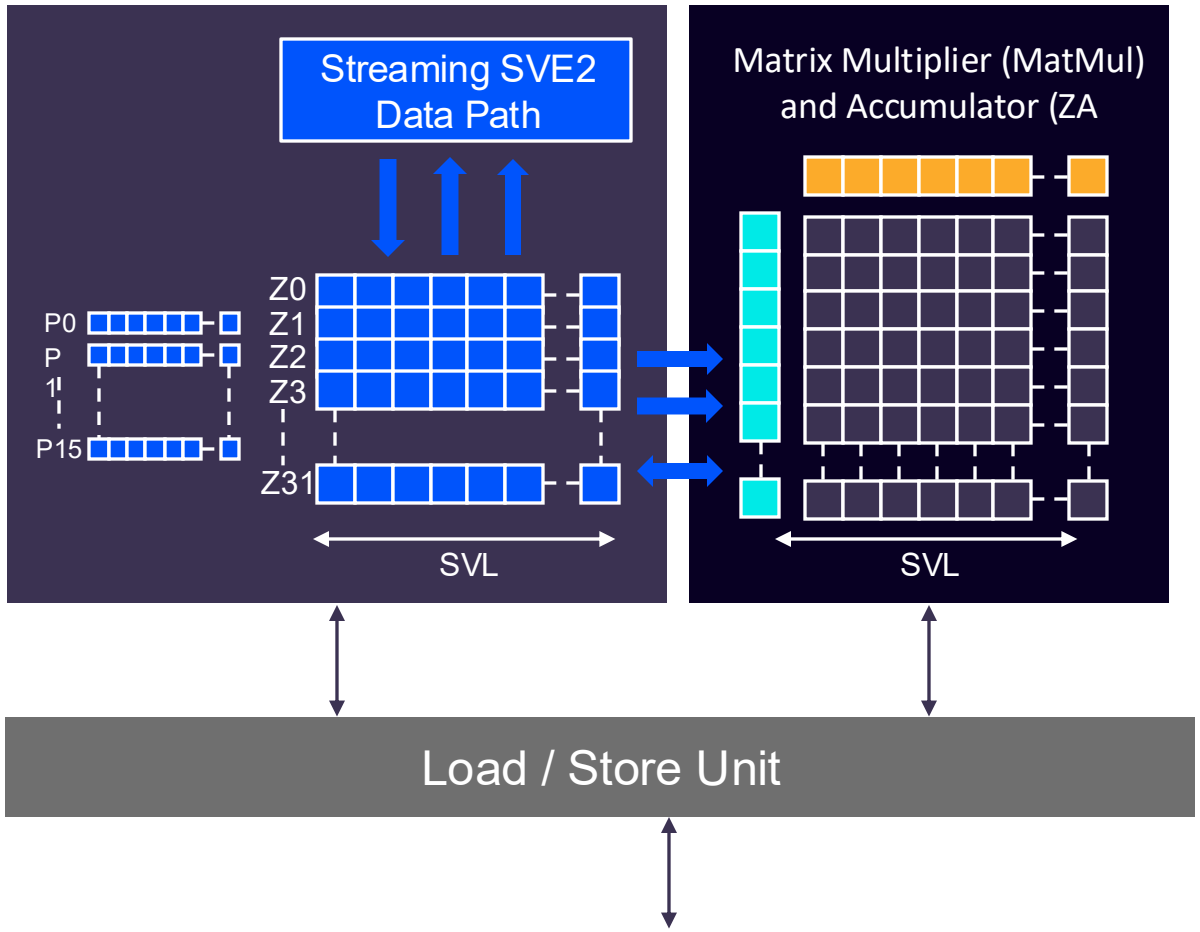


Partial sums are computed as the outer product of a column vector from the left matrix and a row vector from the right matrix

The Scalable Matrix Extension (SME)…

- Matrix processing capabilities with:
  - New vector outer product instructions
  - A 2D array storage ZA to store and accumulate partial sums
- SME extends the Scalable Vector Extensions (SVE and SVE2)
  - Leverages the existing SVE software ecosystem
- Streaming SVE2 enables an implementation to maximize MatMul resources utilization
- Compute capability is made available to software when entering ”streaming mode” (dedicated architecture state). Compute has its own vector length (SVL) which may be different from the VL when not in streaming mode

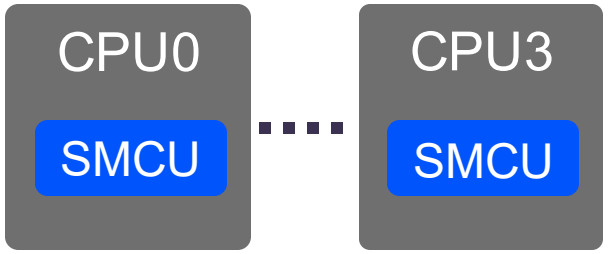
Streaming Mode Compute Unit:



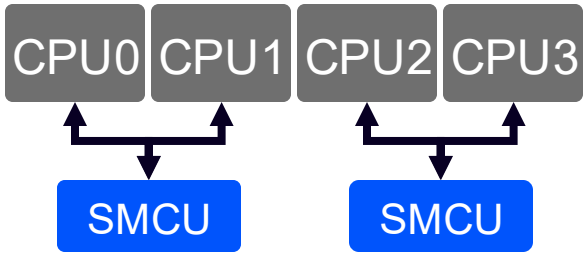
U-architecture Implementation Flexibility: Built-in CPU or Shared

- A Streaming Mode Compute Unit (SMCU) can be shared by many CPUs
  - Different configurations for different trade-offs between cost and performance

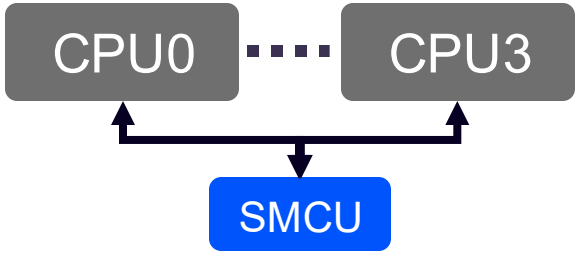
Built in CPU



Multiple shared SMCU



Shared SMCU



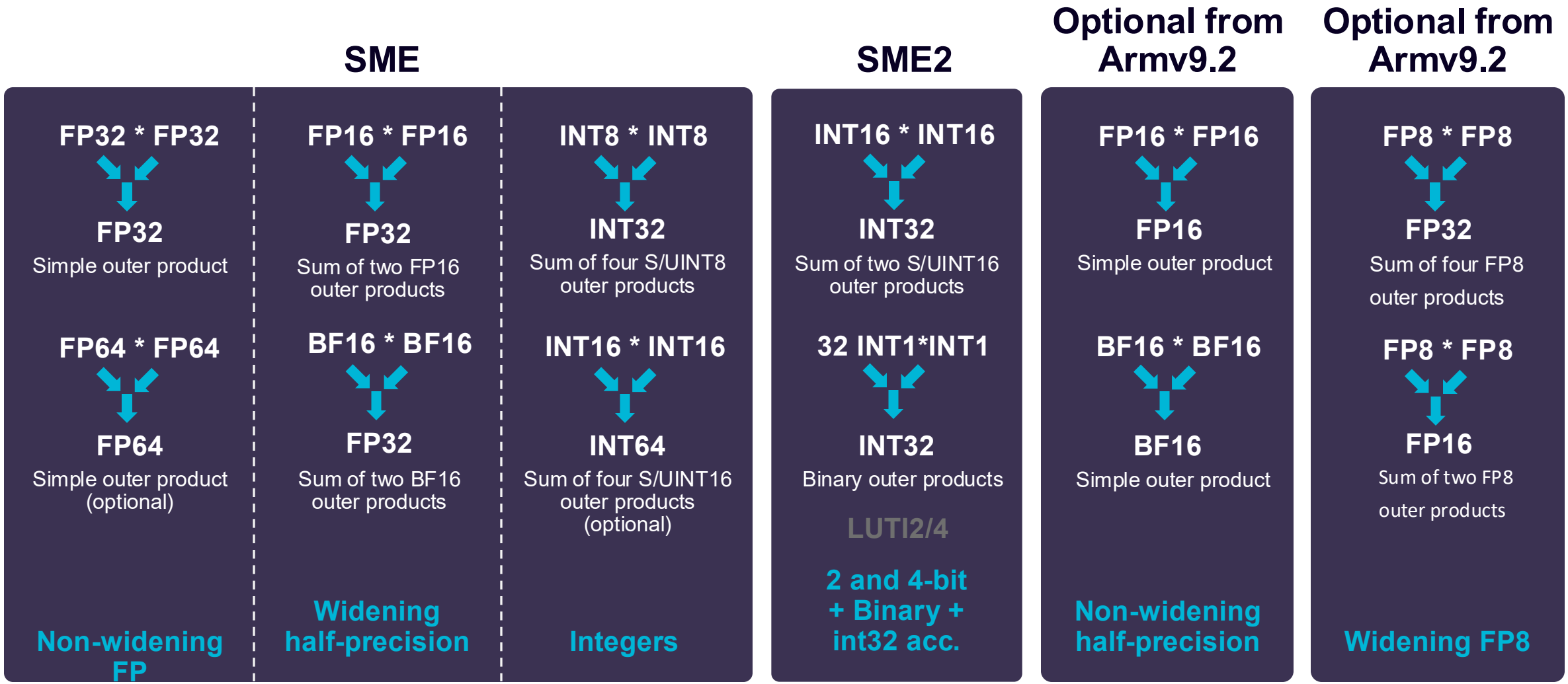
- Favouring multi-threading performance, re-use of In-CPU SVE hardware for SSVE instruction – same VL
- (more) Predictable performance from compute standpoint

- Favouring single-thread performance and performance on low thread count
- Streaming VL does not need to align with the in-CPU (SVE) VL.
- Less predictable performance from compute standpoint

Importantly: Implementation remote from the CPU does not require any software modification – deported compute pipeline.

Flexible Storage and Compute Formats

SME outer product operations – a large set of data types

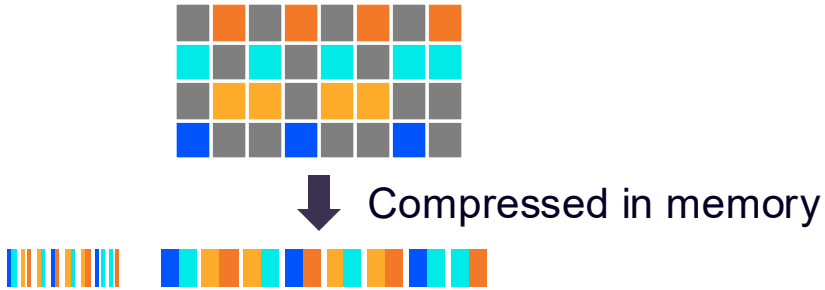


Architecture Extension 2024 (Armv9.6)

Improved performance and flexibility

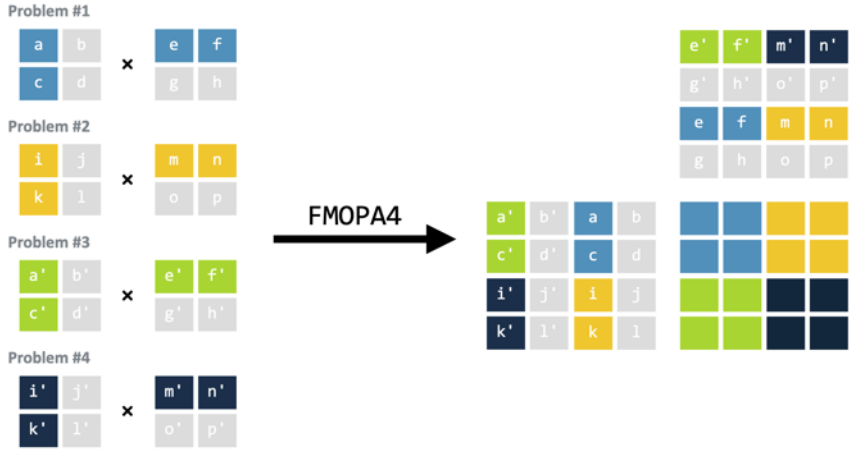
2-in-4 Structured Sparsity Support

- Introduction of 1:2 and 2:4 outer product instructions
- Dense x Sparse operation
- Sparse weights compressed in memory and directly consumed by the instruction
- Compute throughput is doubled



Four independent quarter-tile outer products

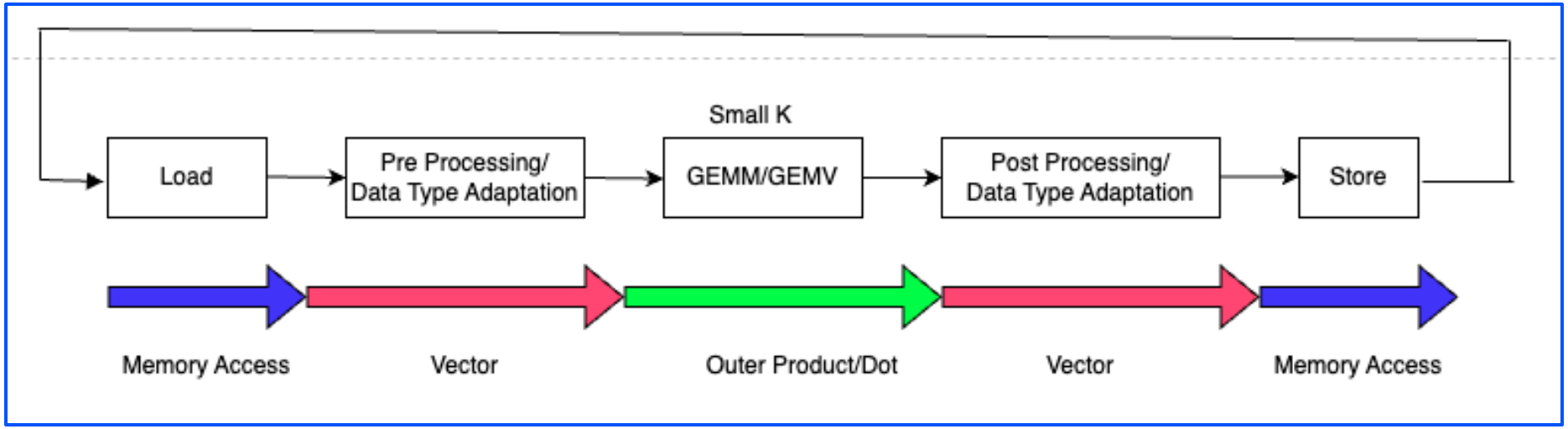
- Improved scalability / better resource efficiency
- Enables significant increase in compute throughput for matrix edges, small number of batches, …



Why is SME more than a dedicated matrix operation engine ?

- SCMU contains both outer-product operation **AND** SVE instructions operating on the same vector length (SVL), because application/kernel are not “only” matrix multiplication:

- Typical Kernel:

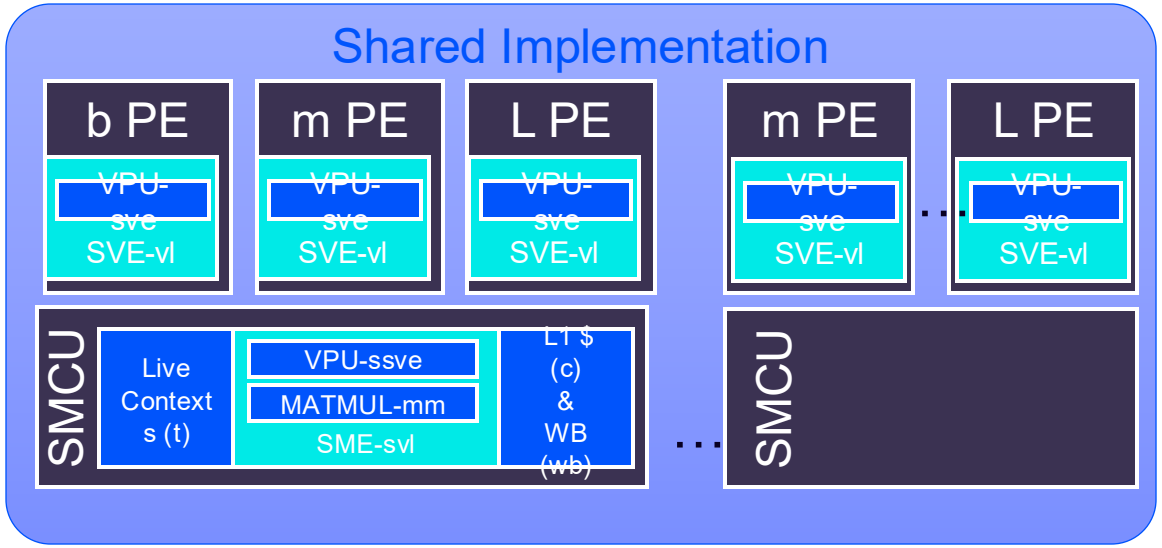


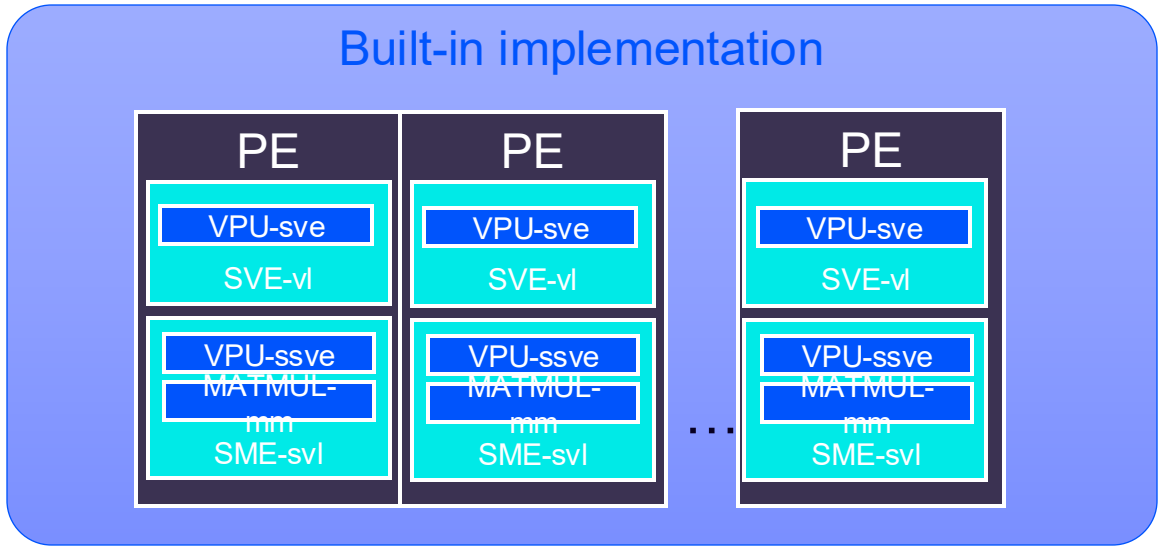
Application	Pre-Processing	Post-Processing
LLM	De-Quantization	Block Level Scaling, Re-Quantization
DFT	Operand Recombination	Twiddle Factor
Structure From Motion/SLAM	Data Rearrangement/Tranposition	Matrix Multiplication border computation (Small-Matrix Computation)

Implementation options and parameters

Architecture State

Implementation





vl	Vector Length
svl	Streaming Vector Length
t	Number of contexts
sve	SVE2 throughput
mm	MatMul throughput
ssve	Streaming SVE2 throughput
wb	SMCU Write Buffer size
c	SMCU L1/L2 Cache sizes
clk_ratio	Clock ratio between CPU and SMCU
smcu	Number of SMCU
pe	PE per SMCU
bw	L1, L2, L3 , Interconnect and DDR Bandwidths
	Homogenous vs heterogenous (Big, medium, LITTLE

Scalable Matrix Extension – Use-Cases

Use-Case A: AI and Generative AI

Generating new content

Natural Language Processing

Chat bot / LLM, translation, summarization, sentiment analysis, …

Audio

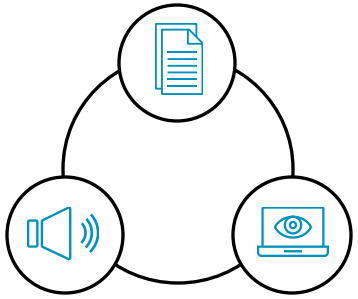
Speech recogniction, Voice control, subtitles, Audio/Speech generation, …

Computer Vision

Image classification, object detection, image segmentation, image retrieval, depth estimation, …

Image/Video

Image generation, Video generation, 3D scene generation, …



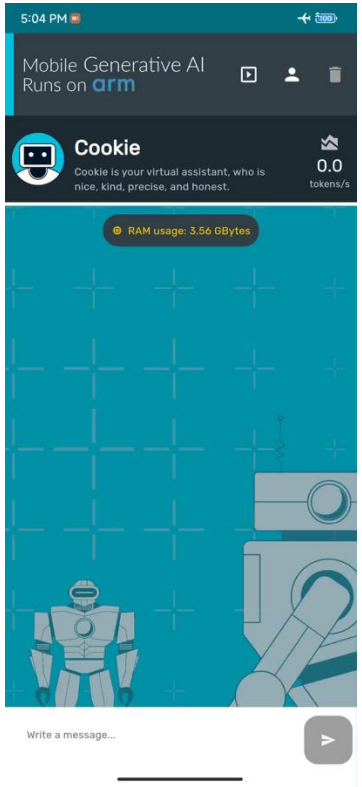
Multi-Modal

Image generation

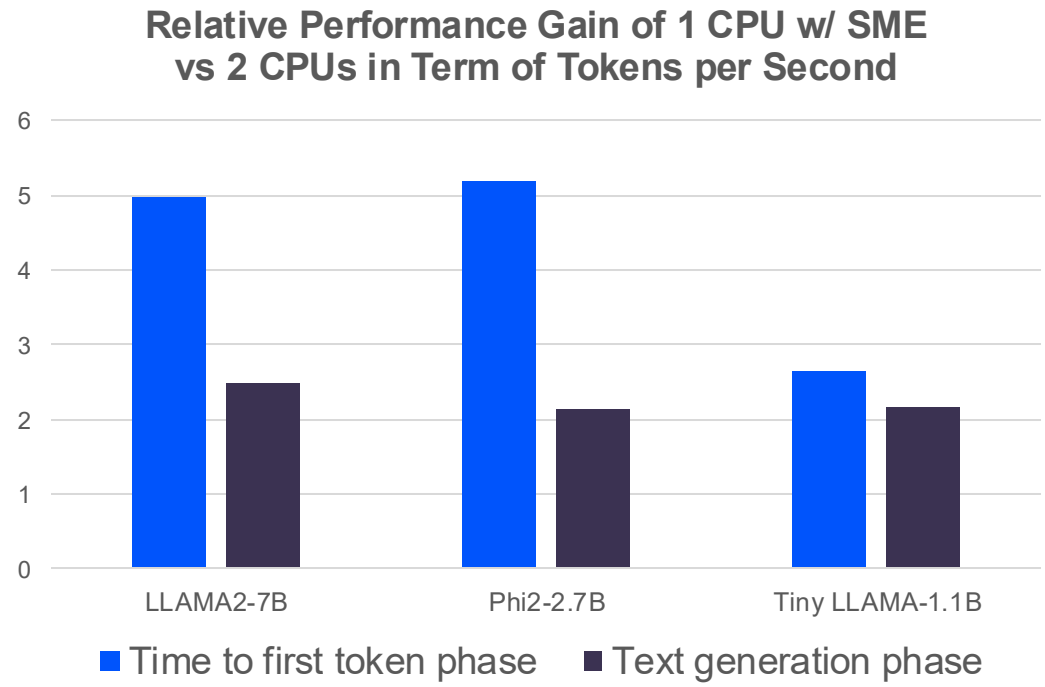
Music generation

SME2 Acceleration of LLM –

- Chatbot use case corresponding to 256 input tokens – 256 output tokens



- SME2 kernels in llama.cpp framework
  - 4-bits quantized GEMM & GEMV
  - Batched strided fp16fp32 GEMM & GEMV

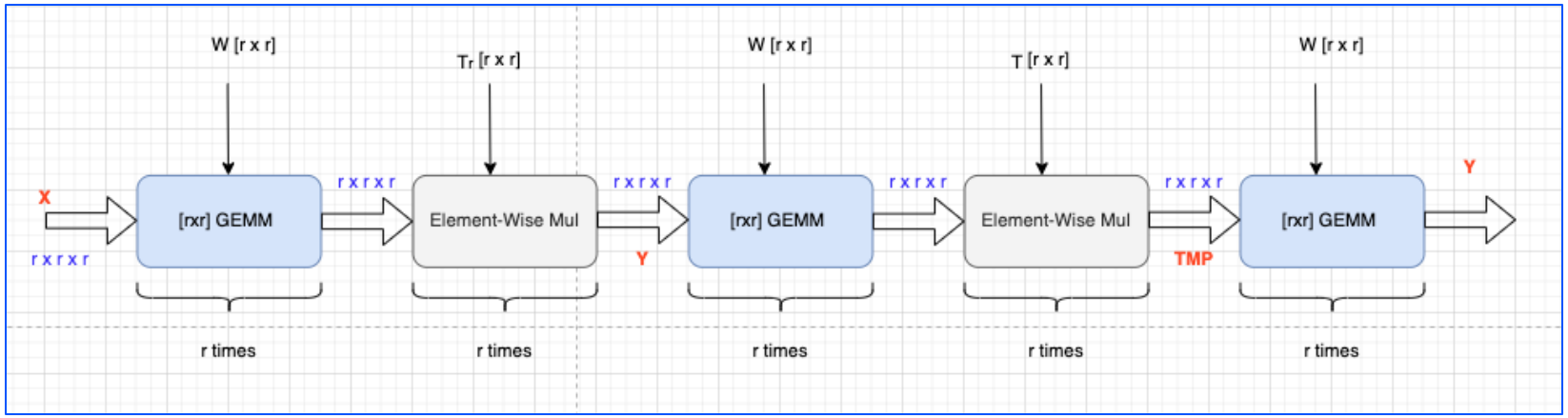


Figures based on architecture performance model

Use-Case B: Discrete Fourier Transform (DFT)

- General Idea:
  - Leveraging SME - Not a given …
  - Need to re-shuffle state-of-the-art DFT computation flow to exhibit outer product operations.
  - Underlying trade-off consists in increasing the number of basic mathematical operations (== fewer intermediate results re-use) but ensuring that those operations better fit outer product operations (which is accelerated by SME)

- Outer-Product based Implementation: Radix size (r) – DFT r x r x r elements:



-> 3*r GEMM operations [r x r]*[r x r]

-> 2*r elements-wise products operations of size [r x r]. r elements-wise operations are done with the same T[r x r] twiddle matrix while the r other element-wise operations are done with different Tr[r x r] twiddle matrices

DFT – Radix16 Accuracy & SME Performance

- Radix-16 has been selected for this proof of concept – selecting SME SVL 512
- Radix-16 is bounded to be less accurate than lower radix-based DFT. Two complex-float32 cases have been analysed – DFT-4096 and DFT-2D 256x256

fftw_float64	4.849 e-16
fftw_float32	1.98 e-7
Radix16-float32 – SME	2.38 e-7

DFT-2D 256x256

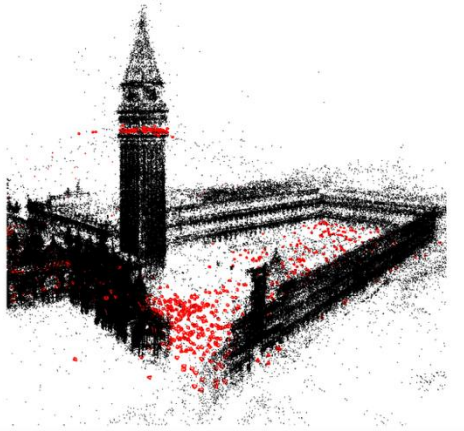
fftw_float64	4.1 e-16
fftw_float32	1.7 e-7
Radix16-float32 – SME	2.06 e-7

DFT-1D 4096

- Simulation comparing SMCU-512 vs a typical CPU containing two vector units – SVE-128:

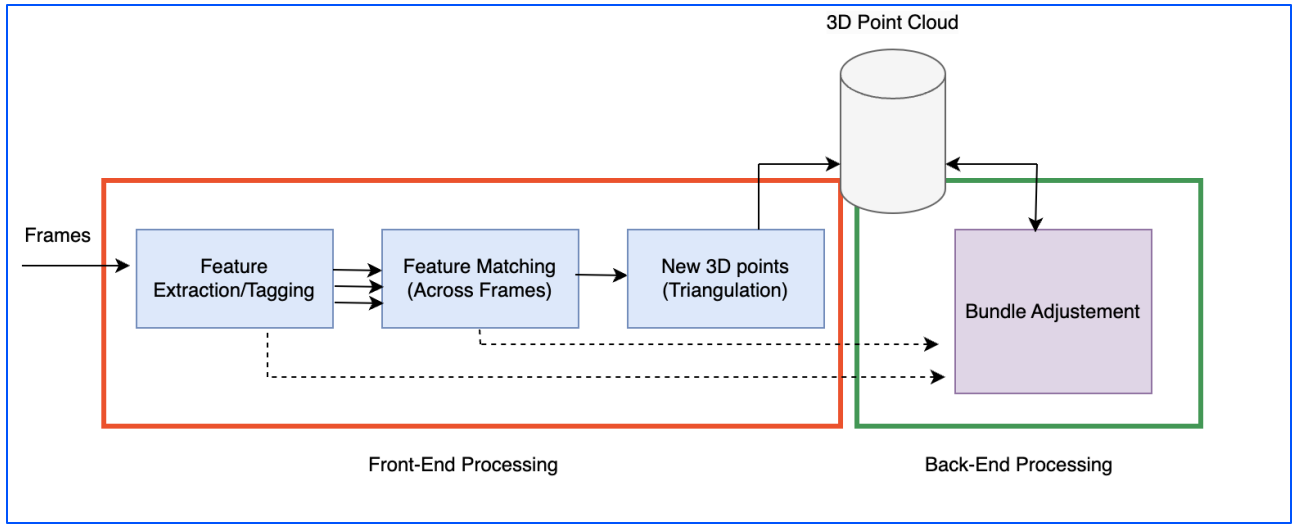
	SMCU Performance vs CPU
DFT-4096	> 7x
DFT-2D 256x256	> 8.5x

Use Case C: Bundle Adjustment (BA)



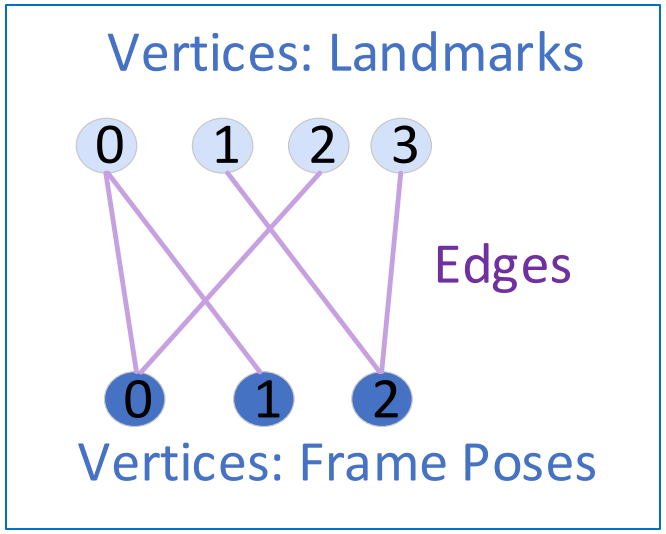
- Building/maintaining a 3D-cloud point from a collection of images using BA technique:

- Bundle Adjustment algorithm is an essential part of most Augmented Reality (SLAM) and 3D reconstruction (SfM) applications techniques.

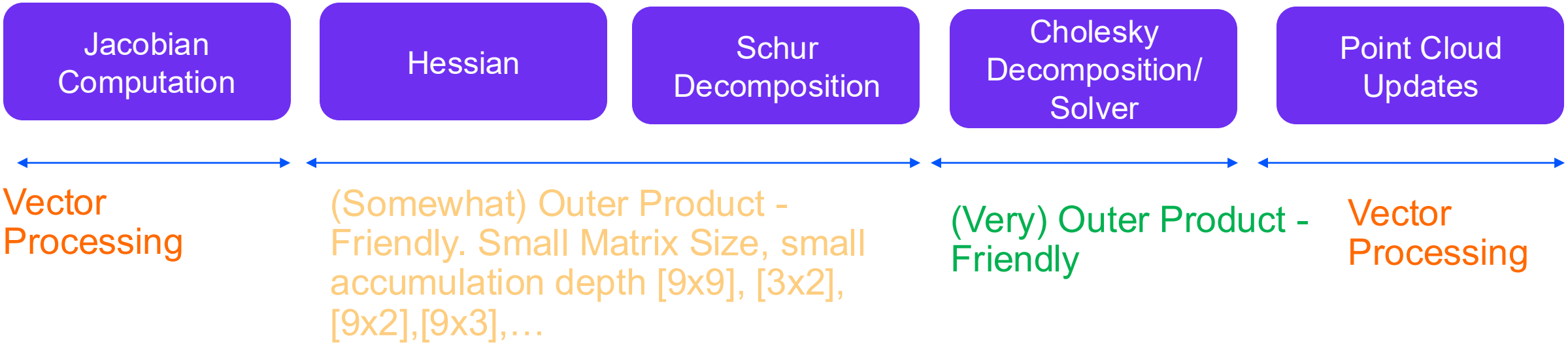


- Front-end processing is aiming at extracting ”features” (corners) from image and performing matching
- Back-end (BA) performs a multi-frame combined camera pose/attributes and point cloud positions optimization (essential for accurate localization or 3D reconstruction).

Bundle Adjustment: High-Level Principle.



- Graph Optimization problem:
  - Optimize camera poses – Bottom Vertices - as well as 3D point cloud (landmarks) position- Top Vertices - minimizing the reprojection errors – Edges
  - Reprojection error: distance between 3D point reprojection onto a given image and its observed trace (ORBSLAM2, BAL, …)
  - Some implementations are aiming at reducing the photometric error.



SfM – SME / Eigen (NEON/SVE) Performance Boost

Simulation: Assuming SMCU-512 vs a typical CPU with 4 vector units of VL 128

Flavor	Camera Parameters	Graph (poses/landmarks/density)	Gain (single thread)
BAL Format /FP64	9	Venice (BAL): 52/64K/10%	> 4x
	9	100/5000/10%	> 5x
	9	100/5000/20%	> 6x
	9	100/5000/30%	> 6x
	9	100/5000/40%	> 5x

- Problem sizing is impact the benefit of SME usage
  - Cholesky/Solver: Number of poses
  - Schur Decomposition: Density

Conclusion & Summary

- Arm has introduced a new matrix extension (SME) which provide significant flexibility for its implementation – ranging from vector length, shared/non-shared implementation,…
- Implementation can adjust compute throughput dedicated to vector pipelines versus matrix multiplication compute throughput based on their targeted application
- Application that can benefit SME are numerous – for instance:
  - (Obviously) those relying on large GEMM computes – genAI, Dense Cholesky, …
  - But also those relying on smaller GEMM and/or tightly coupled interaction with vector processing – Linear Algebra based algorithm relying on iterative linearization/solver, DFT
- Yet - some algorithms may need to be rewritten to expose matrix operations – DFT is one example.

arm

Merci
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Thank You
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు
Köszönöm