RISC-V Matrix Extension Proposal

T-Head

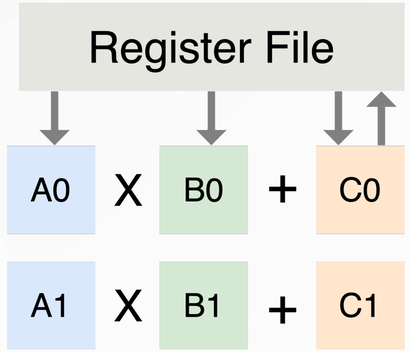
QiuJing
July 2023

Contents

Why Matrix Extension

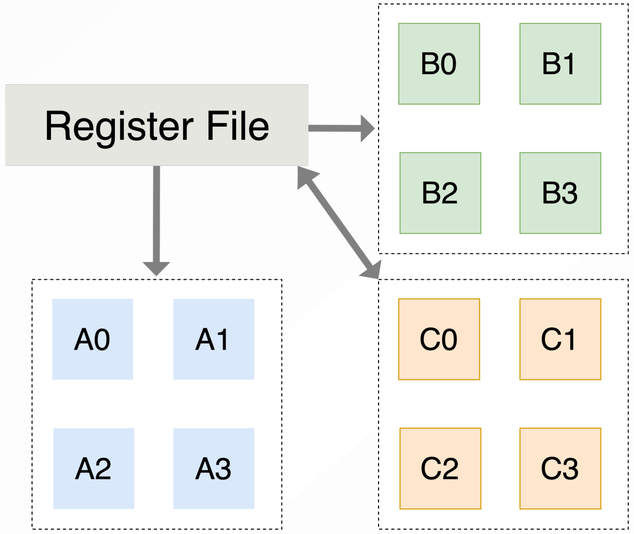
vector

N elements --- N operations



Matrix

N^2 elements --- N^3 operations



AI domain specific architecture

Performance improvement

2x – 8x performance boost

Reduced memory bandwidth requirement

less data to generate more operations (data memory bandwidth)

one instruction contains N^3 operations (inst memory bandwidth)

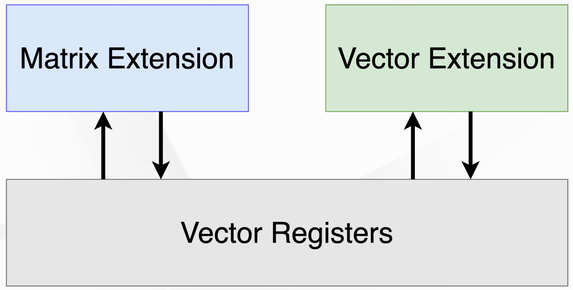
Inherently more power efficiency with less data movement

Matrix Extension in CPU

Fully integrated facility

Reuse vector registers for source operands

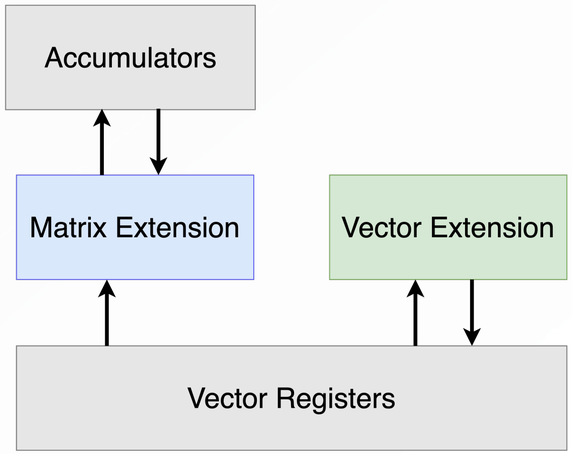
Reuse vector registers for accumulators



Partly attached facility

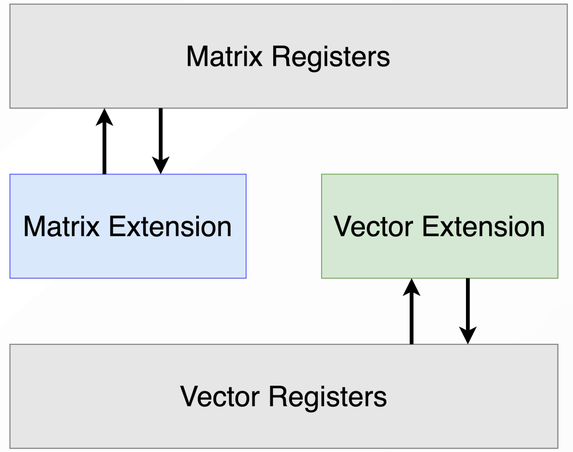
Reuse vector registers for source operands

Independent accumulator registers



Attached facility

Additional matrix registers



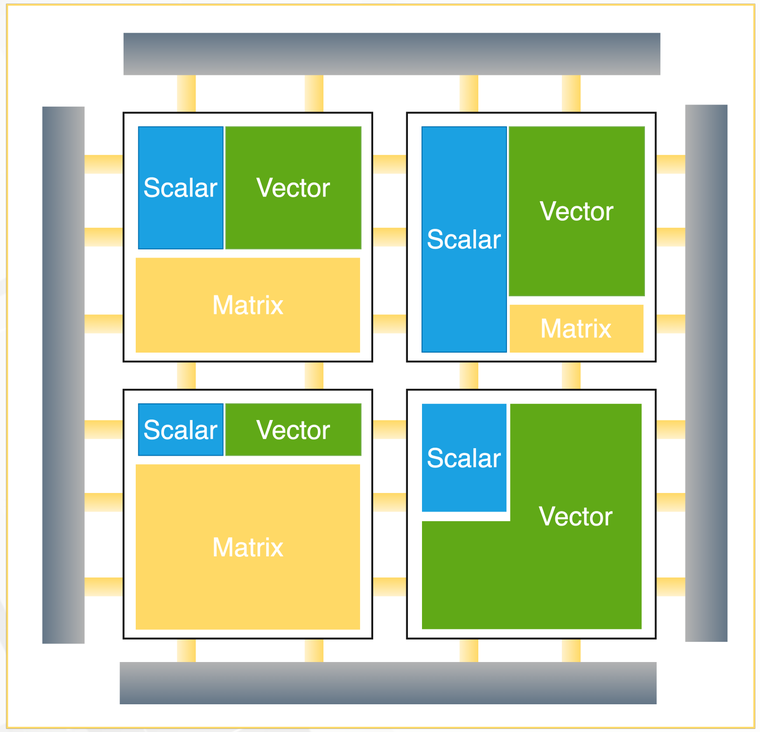
What others do

Reuse vector registers for source operands and accumulators	SIFive Intelligence Extension
Reuse vector registers for source operands independent accumulator register	Power MMA Arm SME
independent source operands and accumulators	Intel AMX Apple AMX Stream Computing Matrix Extension

Why Attached Facility(1/2)

Diversified market needs

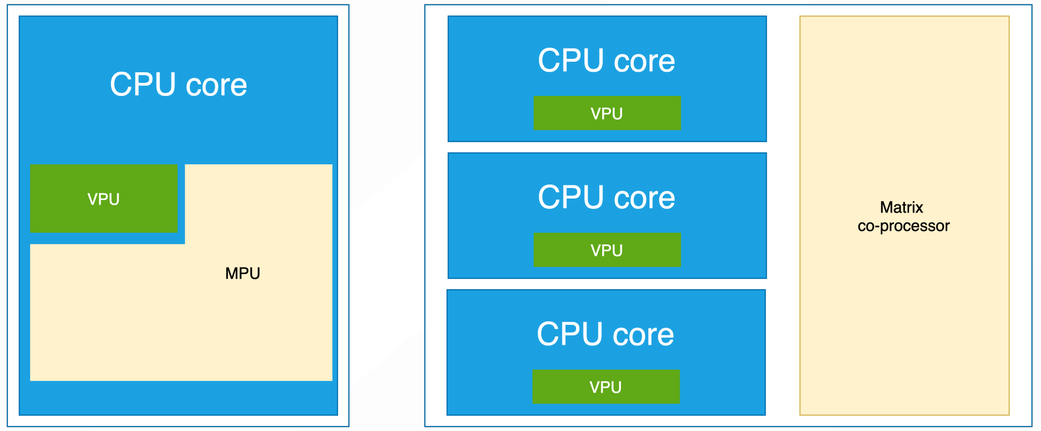
The FLOPs of vector and matrix extension are not always in proportion
(even dropping vector for low power IOT devices)
Matrix and vector operations can be parallel or serial



Hybrid computing structure with different FLOPs proportion

Hardware aspect

Suitable for different implementations,
execution unit/co-processor/DSA/etc.



Only influence matrix circuits when changing AI-related FLOPs
without affecting hundreds of vector instructions
for agile development

Why Attached Facility(2/2)

Developer aspect

Friendly due to separate instructions and programming model

Power aspect

Simplify timing/layout optimization, always better
Separate power design for vector and matrix
IDLE state/voltage supply/DVFS and so on
Separate the heat of vector and matrix

CONS && SOLUTIONS

Resource wasted

Only valid when vector and matrix flops match perfectly
Implementation reuse for low cost core

More data transmission between vector and matrix

Possible with small proportion
Matrix operation instructions to avoid data transmission

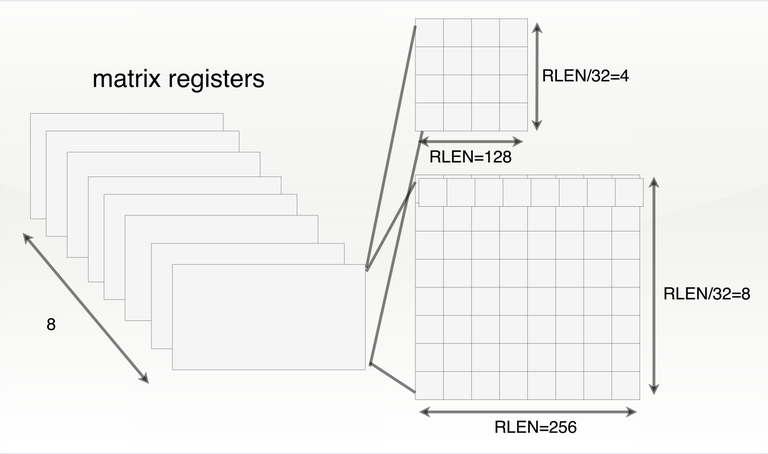
More context switching costs

Dirty flags to save only used registers

Programming Model (GPRs)

8 two-dimensional uniform-size registers

both for source operands and accumulators



RLEN	MLEN	Element Width	Matrix Size
128	512	8	4*16
		16	4*8
		32	4*4
		64	4*2
256	2048	8	8*32
		16	8*16
		32	8*8
		64	8*4

RLEN the length in bits for each register row

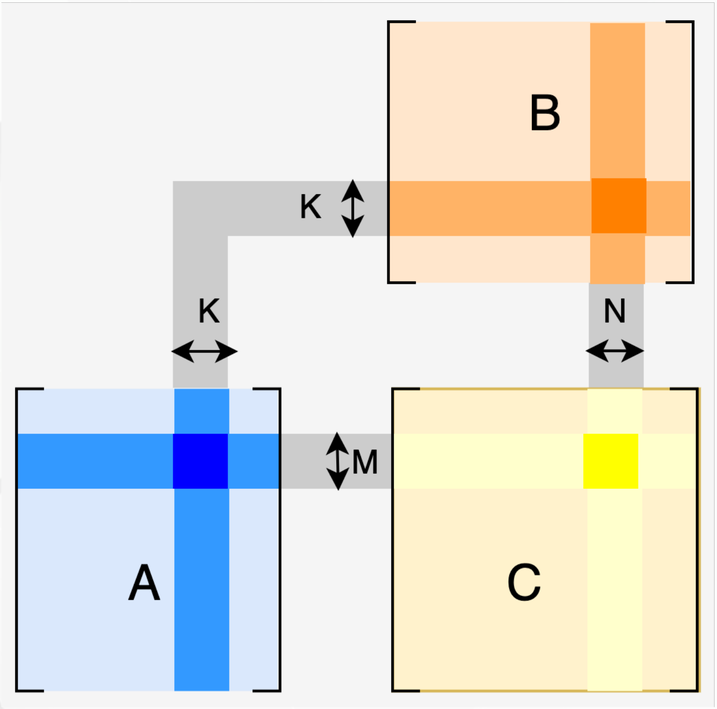
supported RLEN values:128/256/512/1024...

MLEN the size in bits for each register

corresponding MLEN values:512/2048/8192/32768...

Supported Element Widths:8/16/32/64

Programming Model (MACC)



Matrix MACC A x B + C

Integer Multiply-Accumulate Instruction(4x widen)
mmacc.<s/u>.<s/u>.<p/b/h>

Floating-point Multiply-Accumulate Instruction
fmmacc. <h/s/d>

Floating-point Multiply-Accumulate Instruction(widen)
fwmmacc.<h/s>

	suffix	Description
Integer	p	4 bit integer
	b	8 bit integer
	h	16 bit integer
	s	signed
	u	unsigned
Floating-point	b	brain float 16
	h	half precision
	s	single precision
	d	double precision

Supported data types int4 int8 int16 fp16 bf16 fp32 fp64

Programming Model (ISA)

Matrix MACC	fmmacc.<h/s/d>	Floating point matrix multiply and accumulate
	fwmmacc.<h/s>	Floating point matrix multiply and accumulate(widen)
	mmacc.<s/u>.<s/u>.<b/h>	Integer matrix multiply and accumulate(4x widen)
Memory access	mld.<b/h/w/d>	Matrix load to matrix registers
	mst.<b/h/w/d>	Matrix store from matrix registers
	mldm/mstm	Load/store whole matrix register
Matrix operations	madd/msub.<s/d>.<mm/mv/mx>.<x/i>	Matrix add/sub matrix/vector/scalar
	mshift.<s/d>.<mm/mv/mx>.<x/i>	Matrix shift matrix/vector/scalar
	mn4clip.<s/d>.<mm/mv/mx>.<x/i>	Matrix clip
	mmul.<s/d>.<mm/mv/mx>.<x/i>	Matrix cross product matrix/vector/scalar
Move	mmov.mm/mmov.mv.x/mmov.mv.i	Move between matrix registers
	mmov<b/h/w/d>.x.m	Move form matrix register to scalar register
	mdup<b/h/w/d>.m.x/mmov<b/h/w/d>.m.x	Move form scalar register to matrix register
Matrix config	mcfgi mcfg	Matrix tail configuration
Others	release	Release matrix register
	zero	Zero matrix register

Only 20+ matrix instructions

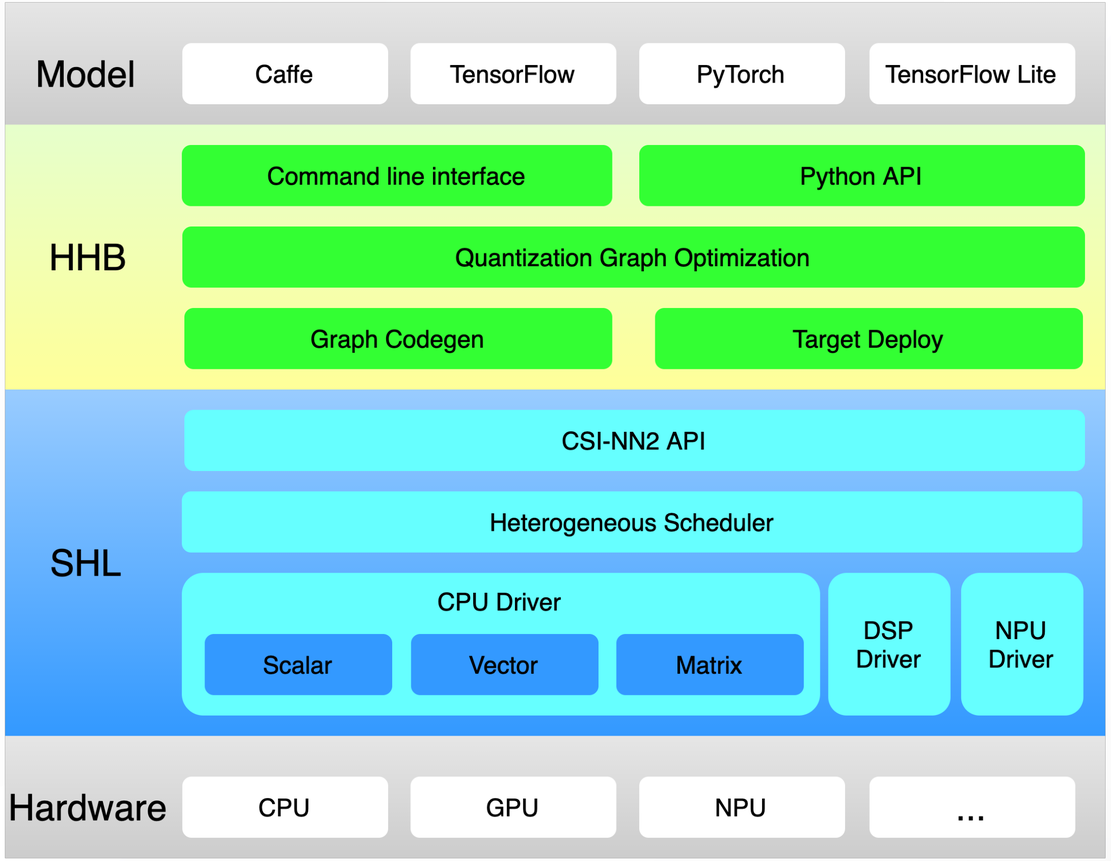
Software support

Compiler and assembler support (GCC and GDB)

Emulator (QEMU)

AI framework

- Deployment toolkit (HHB)
  - Compiler optimization
  - Performance analysis
  - Process debugging
  - Result simulation
- Neural networks library (SHL)
  - Explicit GEMM convolution (Im2col)/Winograd
  - Supported Vector Extension and Matrix Extension



AI Matrix Extension Innovations

AI domain specific

- Small set of matrix multiplication instructions
- Support conv by Winograd/Im2col/Direct algorithms
- AI/ML data types int4/int8/bf16/fp16 and so on
- Multi-precision and mixed-precision computation

Scalability

- RLEN scales from 128 to 1024+
- Peak performance from 0.128 Tops to 32 Tops
- Binary portability no recompile or rewrite for different RLEN
- Various matrix shapes

Attached Facility

- Decoupled from vector extension at programming model level

Extensibility for future

- Future data types(fp4 fp8 binary)
- Other matrix operations(im2col pointwise)
- Other matrix features(sparsity)

Energy efficiency

Matrix operations

- Less data, more flops
- Less instructions, more flops

Matrix extension

- Support for low-precision data types
- Enough registers for data reuse ratio
- Attached structure for low power design
- Reduced instruction set
- Micro architecture optimizations(hint/flags)
- Leave room for future low power design(sparsity/compression)

Contents

Call for Matrix TG

T-HEAD

pre-inception stage

- Invite experts and engineers join the project
- Arrange meetings to discuss about matrix extension and AI applications
- Call for AI matrix extension spec proposals
- Gap analysis for matrix extension

milestone1

- Release matrix extension spec v0.1
- Choose benchmark to value all the proposals

milestone2

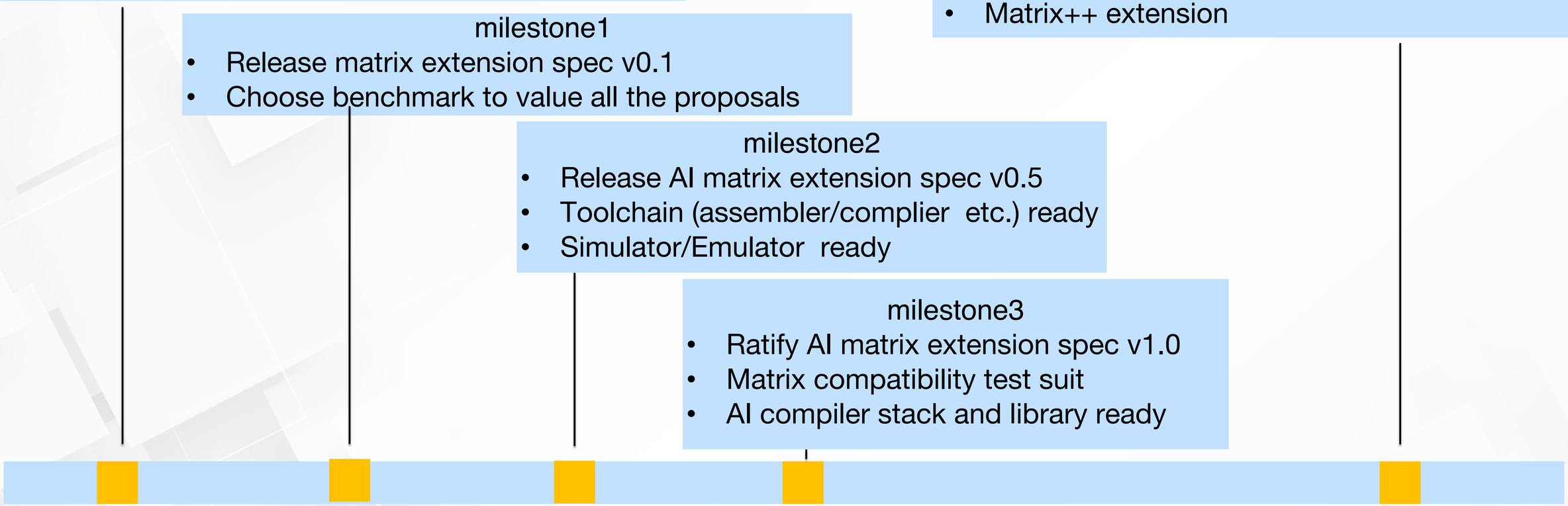
- Release AI matrix extension spec v0.5
- Toolchain (assembler/complier etc.) ready
- Simulator/Emulator ready

milestone3

- Ratify AI matrix extension spec v1.0
- Matrix compatibility test suit
- AI compiler stack and library ready

future plan

- Cooperate with other RISCV group
- OS supported(Linux/Android)
- AI framework supported(TensorFlow/PyTorch)
- Chip released including matrix extension
- Matrix++ extension



Our proposal future works

T-HEAD

More Matrix Extension

- New data type extension (fp8/fp4/binary)
- New operation extension (im2col/pointwise)
- New feature extension (sparsity/compression)

More software ecosystem

- More end-to-end demos
- AI frameworks
- OS supported

Other components

- Silicon Implementation Guild
- Compatibility test suit

Thanks && Questions

more details on https://github.com/T-head-Semi/riscv-matrix-extension-spec

New update features after first released

we are still working hard on it

- Add widen floating-point matrix multiplication instructions for mixed precision
  fp16/bf16 --> fp32 fp32 -->fp64
- Add move instruction between scalar registers and matrix registers for debug purpose
  mmov.m.x mmov.x.m mdup.m.x
- Change mrelease to initial ms status for security issues
- Support new data types int4/bf16
- Add RLEN, modify MLEN definition for a clearer programming model
- Reorganize the matrix CSR to be more accurate and meet the RISC-V standards
- Modify arithmetic instructions opcode, more consistent with RISC-V coding habits
- Remove streaming memory access instruction, Compatible with zhintntl extensions instead of customizing hint operations in extensions
- A few minor bugs fixes
- New intrinsic style with function overloading