平头哥 RISC-V

RISC-V 中国峰会 2023

Matrix Extension for RISC-V Architecture

QiuJing

01 General Intro

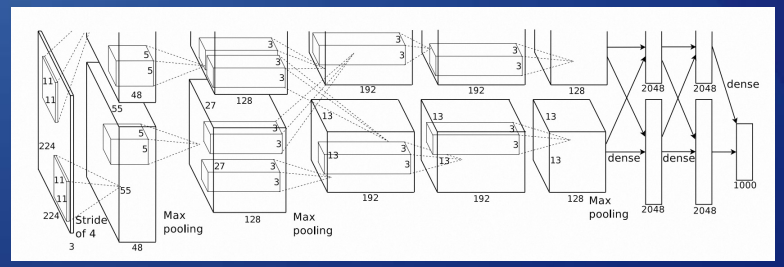
02 Detail Talks

03 Matrix Proposal

04 Future Roadmap

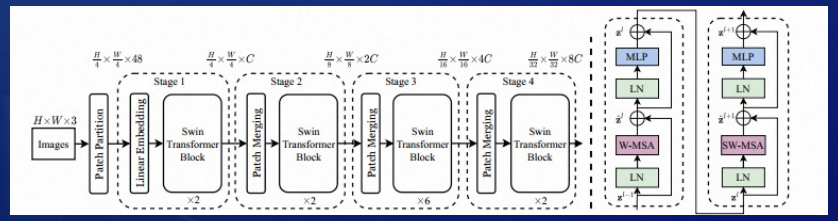
General Introduction

Matrix ISA Extension



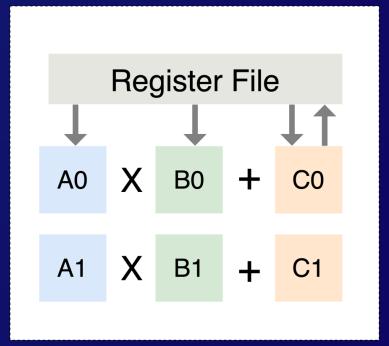
The Architecture of Alexnet

**Imagenet classification with deep convolutional neural networks[J].*



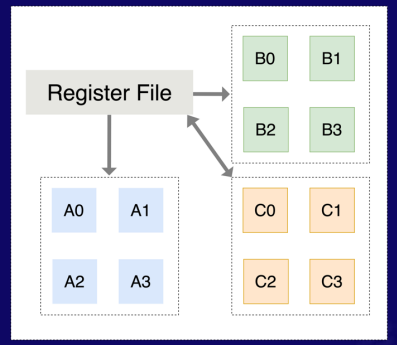
The Architecture of Swim Transformer

**Swin transformer: Hierarchical vision transformer using shifted windows*



Vector

N elements --- N operations



Matrix

N^2 elements --- N^3 operations

Matrix is All You Need

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

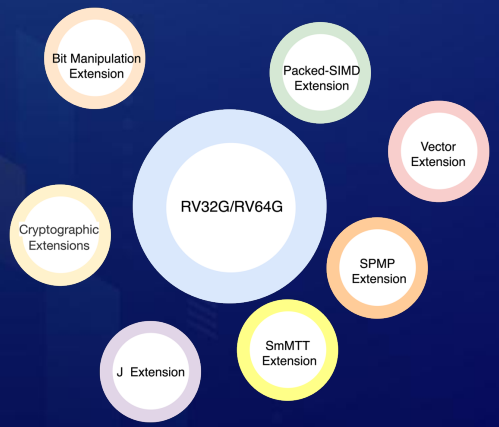
RISC-V Matrix Extension

Open and Free Architecture

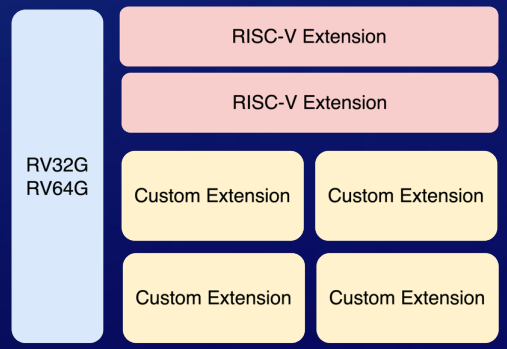
Lightweight and Efficient

Software Compatibility

Modularity



Customize



Goals for RISC-V Matrix Extension

Efficiency

Improve AI performance by accelerating matrix operations

Versatility & Scalability

From low-powered edge devices to high-performance data center

Flexibility

Multiple data types and matrix shapes

Portability

Binary code portable across register size and hardware implementation

Future-Proof

Easily extensible for future AI Matrix++ Extension

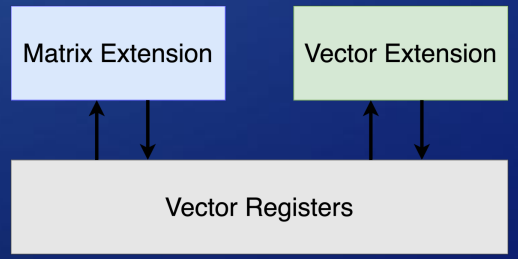
Detail Talks

Matrix in CPU

Fully integrated facility

Reuse vector registers for source operands

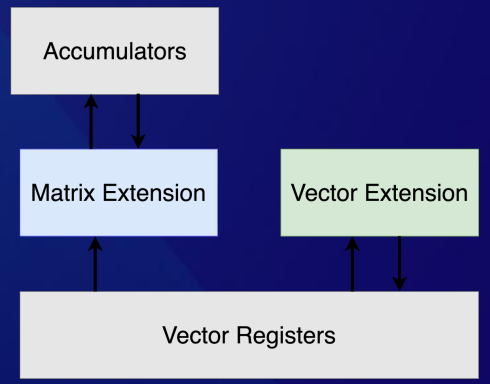
Reuse vector registers for accumulators



Hybrid facility

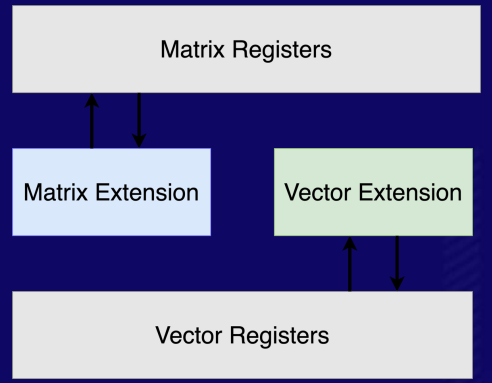
Reuse vector registers for source operands

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

Attached Facility Matrix Extension

Diversified market needs

Parallel processing

Implementation Flexibility

Agile development (both hardware and software)

Power Optimization

Resource wasted

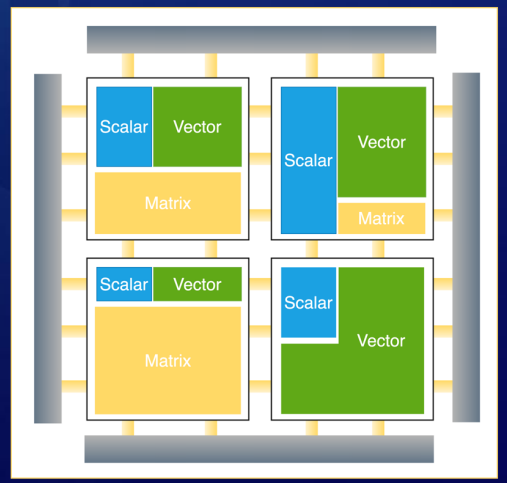
Implementation reuse for low cost core

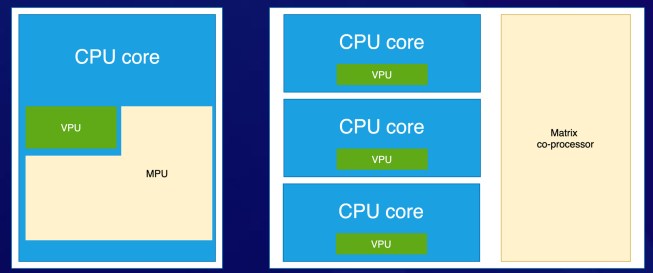
More data transmission between vector and matrix

Matrix operation instructions to avoid data transmission

More context switching costs

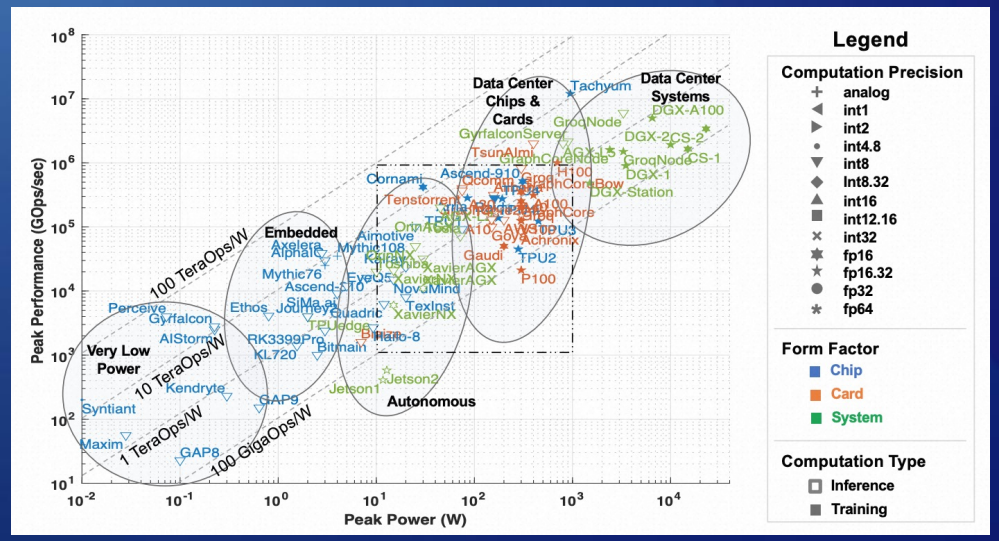
Dirty flags to save only used registers





Vector Extension	Matrix Extension
Hundreds of instructions	Only 20+ instructions

Data Types



Variety of data types

** AI and ML accelerator survey and trends*

MATRIX_MULT_F32F64	optional
MATRIX_MULT_F16F32	optional
MATRIX_MULT_F8F16	optional
MATRIX_MULT_F8F32	optional
MATRIX_MULT_F64F64	optional
MATRIX_MULT_F32F32	optional
MATRIX_MULT_F16F16	optional
MATRIX_MULT_I16I64	optional
MATRIX_MULT_I8I32	compulsory
MATRIX_MULT_I4I32	optional
OTHERs	optional

MATRIX_MULT_I8I32

MATRIX_MULT_I4I32

MATRIX_MULT_F16F16

MATRIX_MULT_F8F16

Device Inference Chip

one example

compulsory + optional data types

- Flexibility
- Efficiency
- Accuracy
- Mixed-precision
- Compatibility

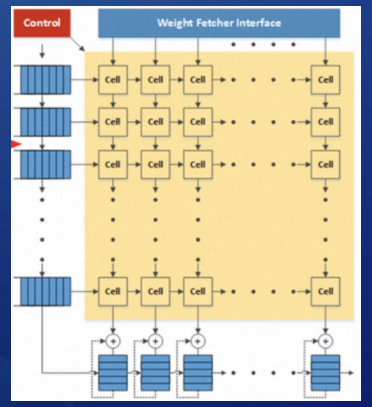
Different data types with same length

bf16/fp16 fp8 E4M3/E5M2

Self-contained

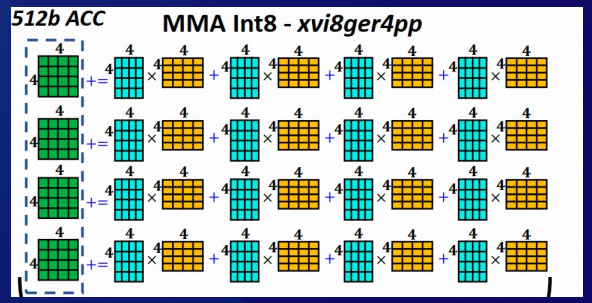
☑ CSR-contained

Matrix Sizes



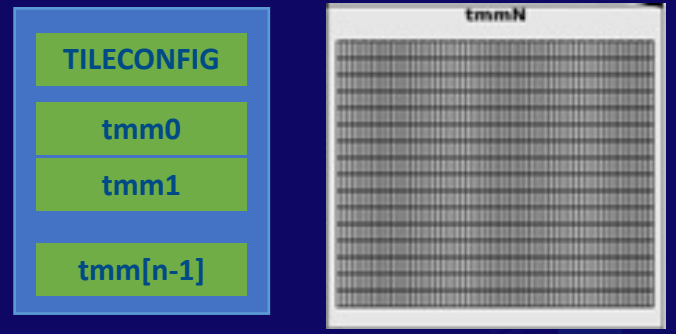
TPUv1 1 256*256
TPUv2 1 128*128
TPUv3 2 128*128
TPUv4i 4 128*128

*Hardware Acceleration of Explainable Machine Learning using Tensor Processing Units
*Ten Lessons From Three Generations Shaped Google's TPUv4i



MMA 4*4

*Server_Processors_IBM_Starke_POWER10

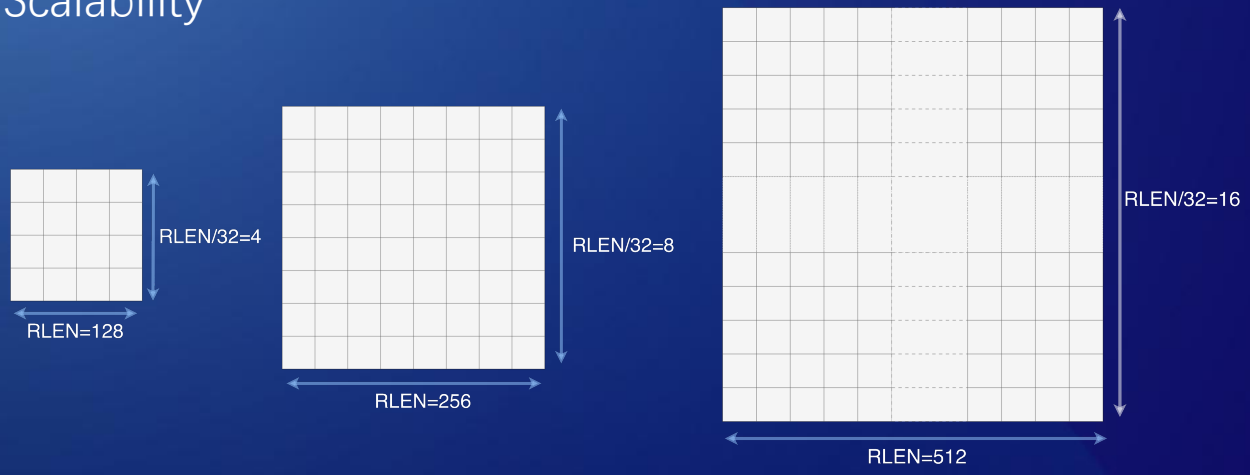


AMX 16*64

Intel® AMX & TMUL Enabling in Linux Kernel

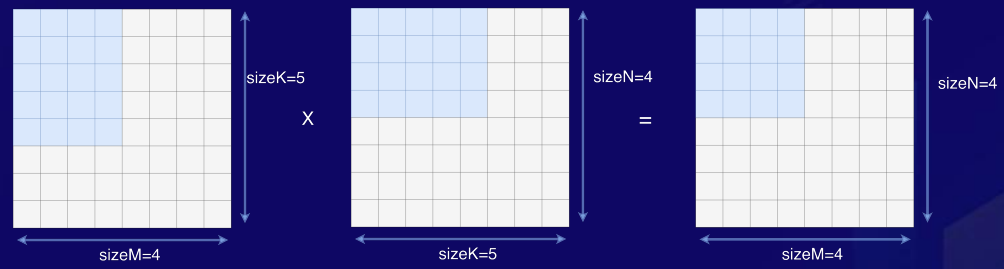
Matrix Shapes

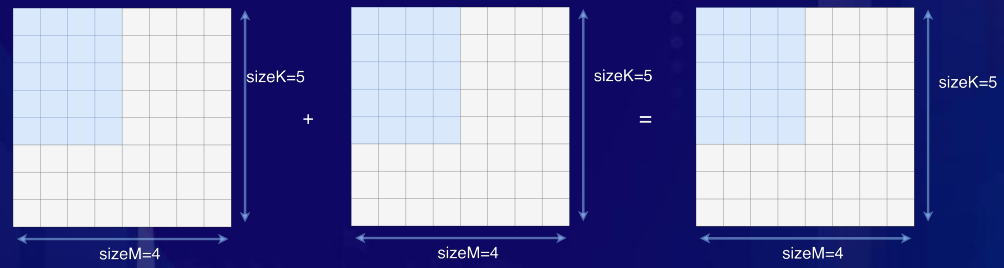
Scalability



Tail configured

sizeK	Matrix A columns
sizeN	Matrix B columns
sizeM	Matrix A rows

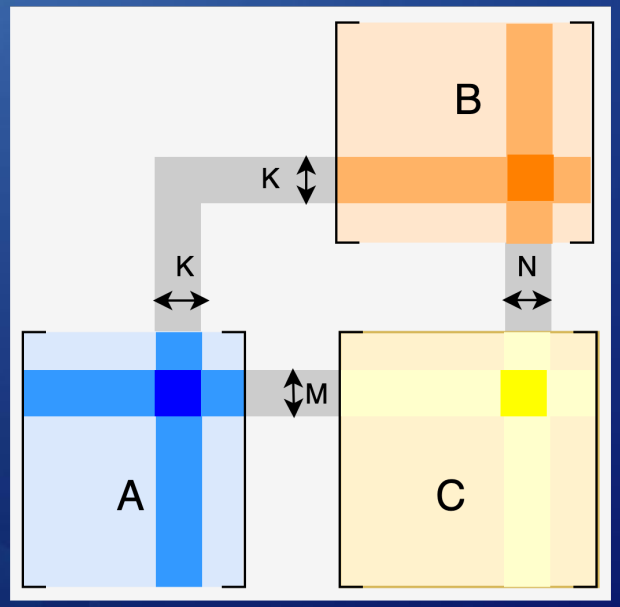




Row fixed

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

Matrix Multiplication



uniform-sized source and accumulation

Non-widen matrix multiplication

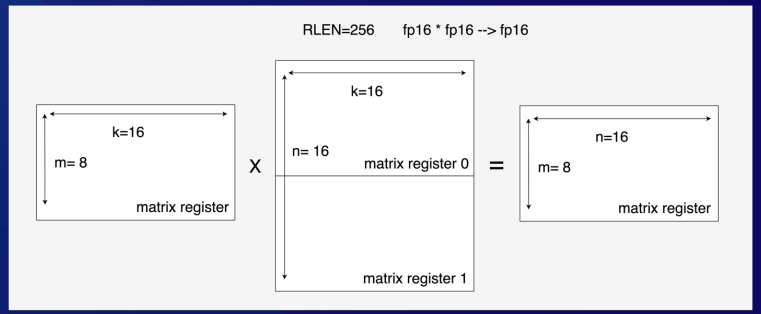
Widen matrix multiplication

Widen(4x) matrix multiplication

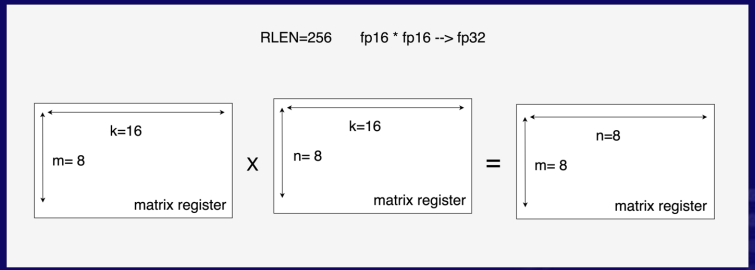
Dense matrix multiplication

Sparse matrix multiplication

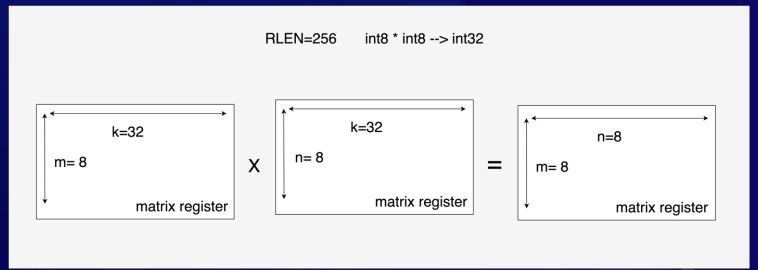
Non-widen Dense



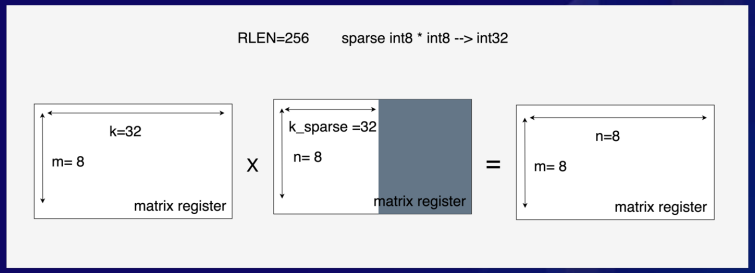
Widen(2x) Dense



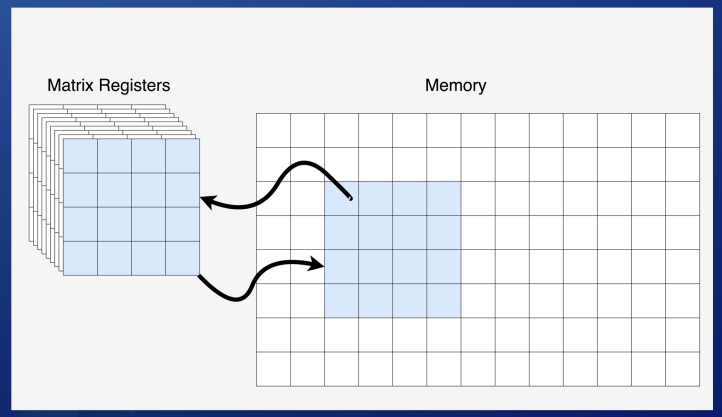
Widen(4x) Dense



Widen(4x) Sparse



Matrix Memory Access



Compatible with zhintntl Non-Temporal Locality Hints

Load multiple rows of contiguous bytes/halfwords/words/doublewords to matrix registers. It does not exhibit temporal locality of private cache

NTL.P1
MLD md,rs2,rs1

Store multiple rows of contiguous bytes/halfwords/words/doublewords from matrix registers. It does not exhibit temporal locality of cache

NTL.ALL
MST ms3,rs2,rs1

Whole Register Load/Store

save and restore matrix registers when the types or length of the current contents of the matrix register is not known

MLDM md,rs1
MSTM ms3,rs1

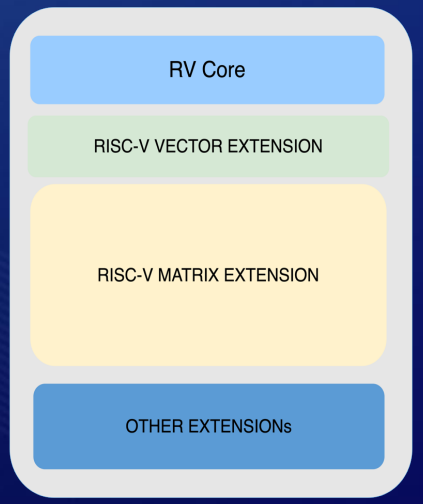
Matrix Proposal

Our Proposal

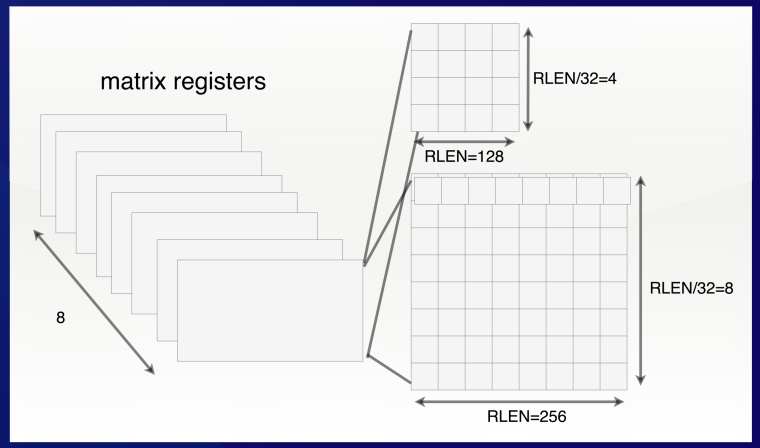
Aim to improve AI inference/training performance

attached facility matrix extension

without or with small/large vector extension



8 two-dimensional uniform-size registers
both for source operands and accumulators



RLEN	MLEN	Element Width	Matrix Size
256	2048	4	8*64
		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...

EW
element width
supported EW values
4/8/16/32/64...

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	Matrix tail configuration
	mcfg	
Others	release	Release matrix register
	zero	Zero matrix register

Only 20+ matrix instructions

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

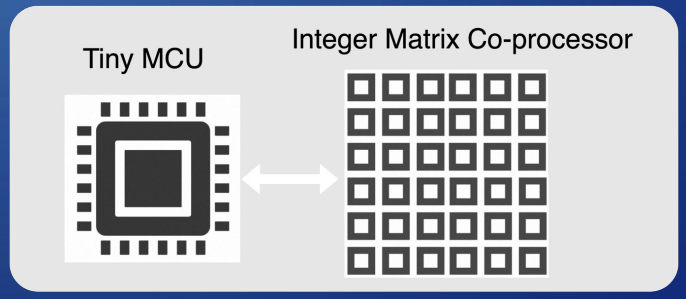
- Less data, more flops
- Less instructions, more flops

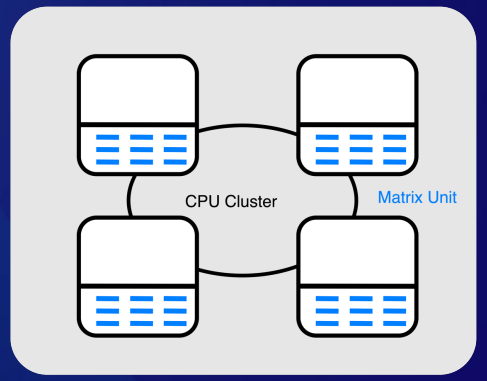
Matrix extension

- 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)

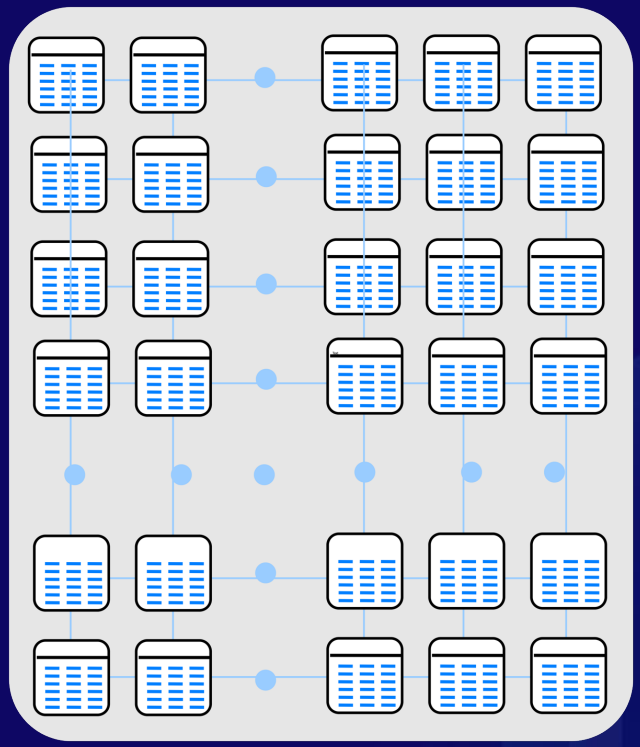
Future Roadmap

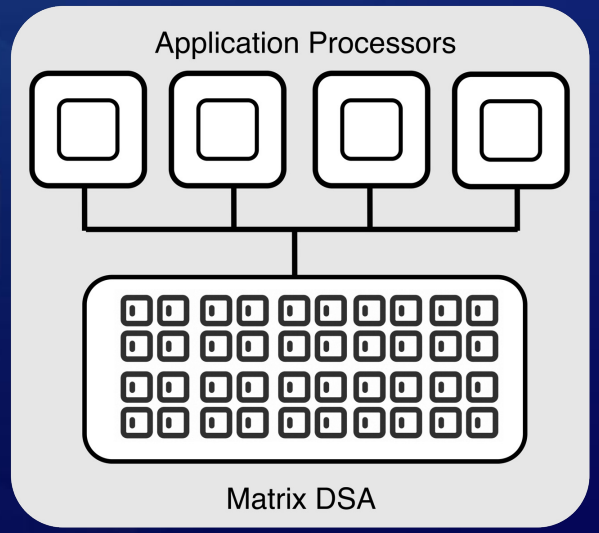
Implementations

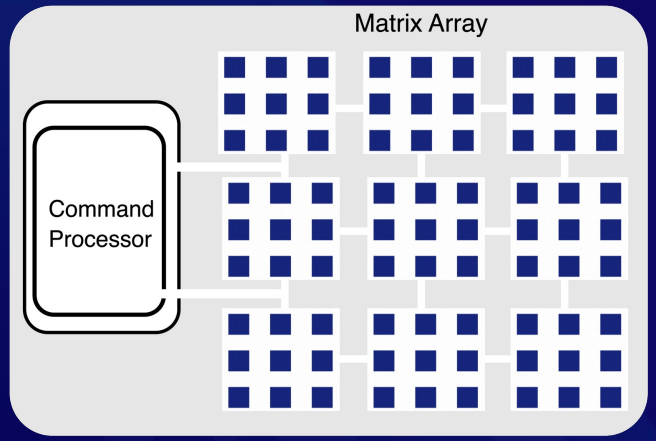




systolic matrix processor array







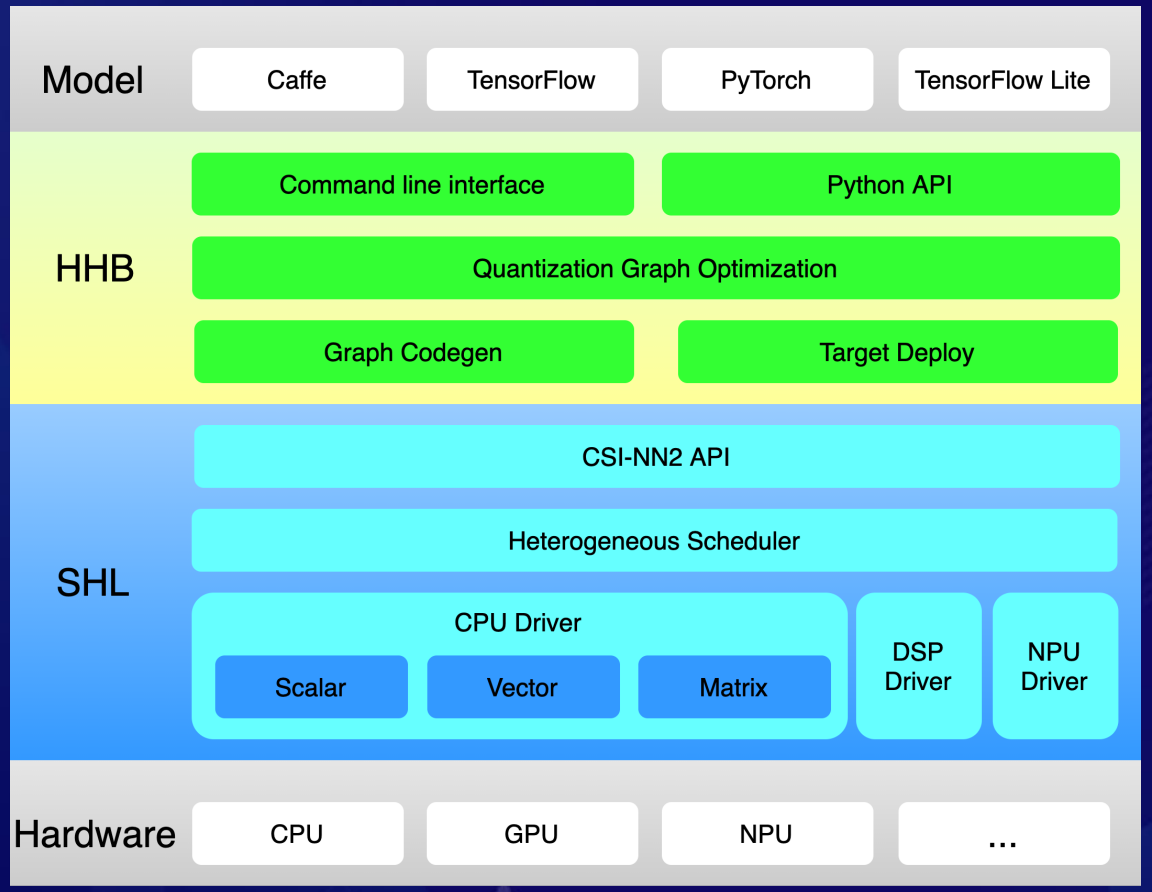
Softwares

More software ecosystem

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

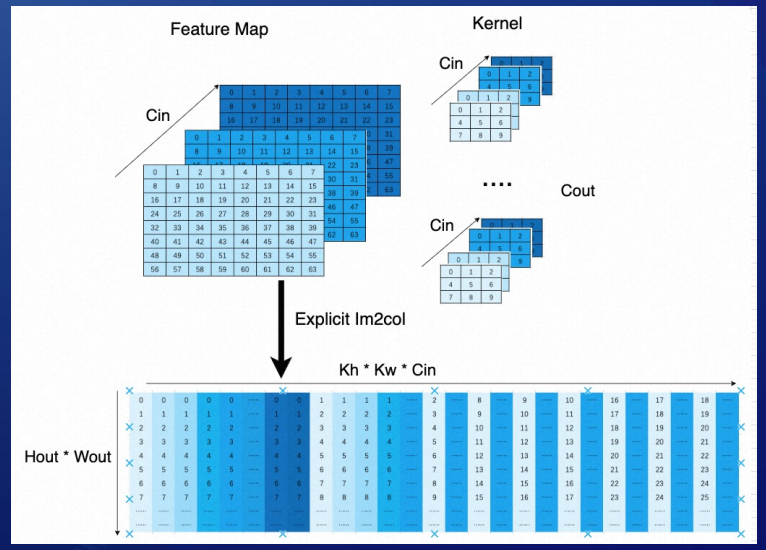
Other components

- Silicon Implementation Guild
- Compatibility test suit

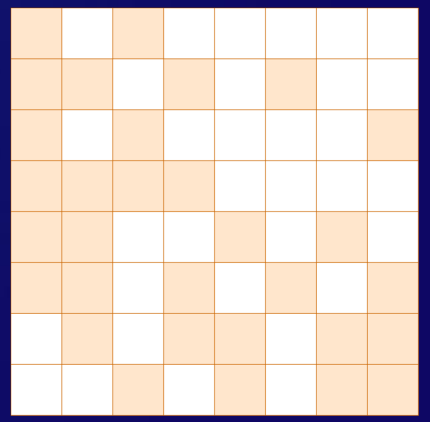


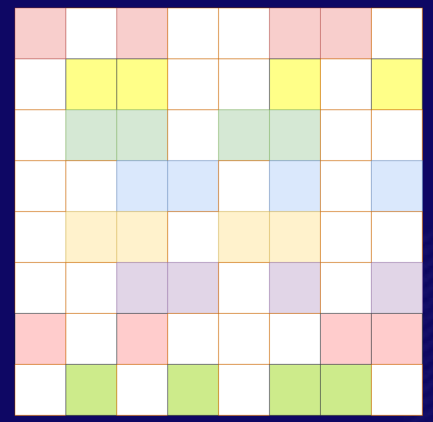
Matrix+ Extensions

New operation extension (im2col/pointwise)



New feature extension (sparsity/compression)





New data type extension (fp8/fp4/binary)

Find More

Xuantie @
GitHub