Scalable Matrix Architecture (Integrated Facility Variant)

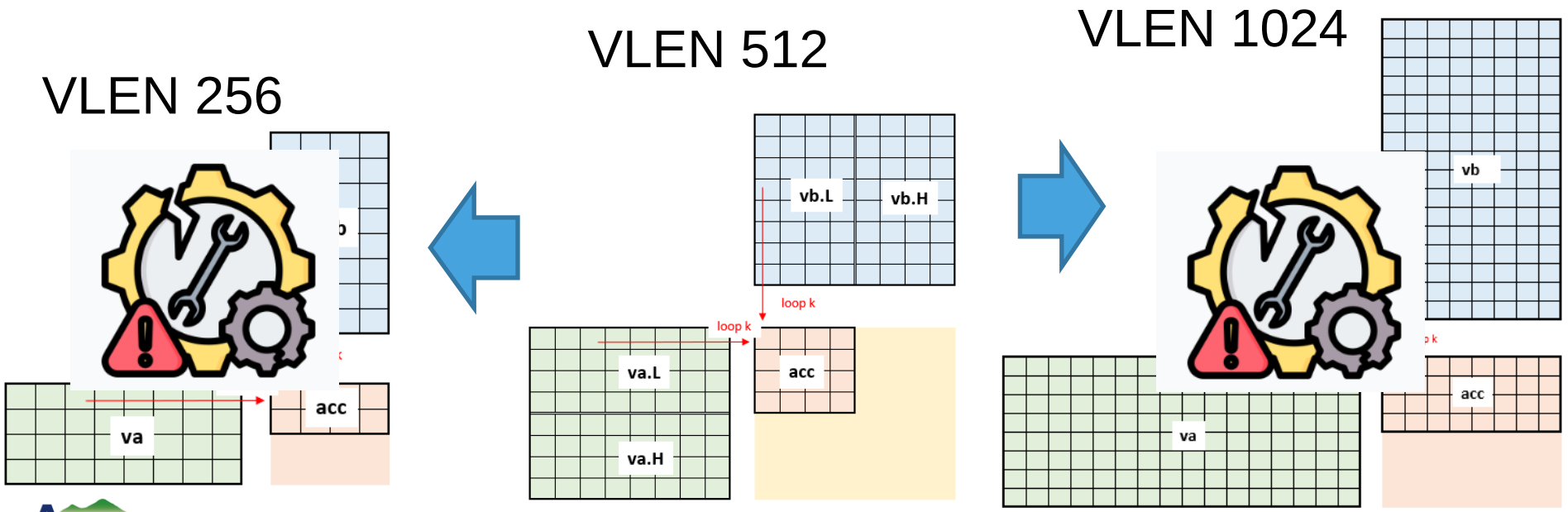
Jim. CN.Ke
Andes Technology

Agenda

- SW Scalability
  - Integrated Facility Challenges Recap
  - Scalable Program Model
  - Optimal Program Model
- Widening Support
- Register Group Support (lmul)
- Summary

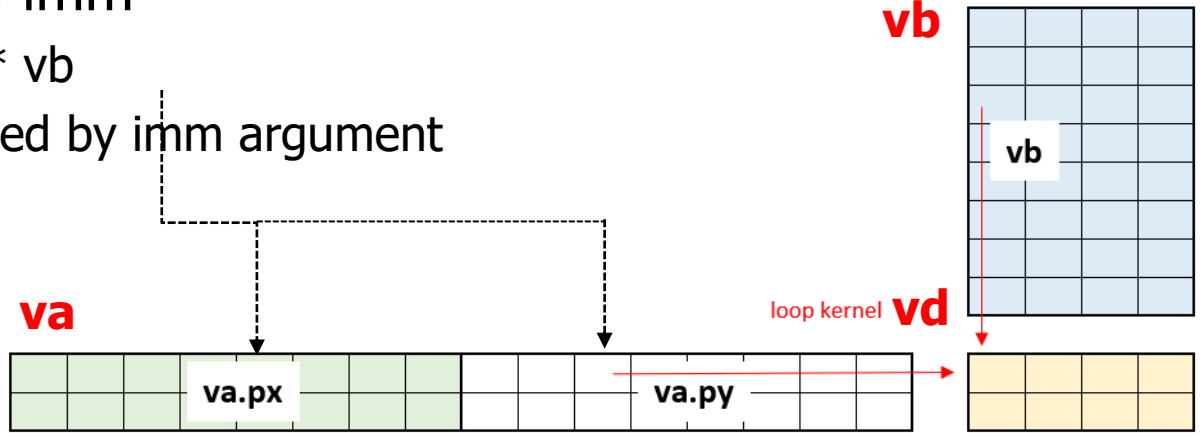
SW Scalability Challenges

- Tile Shape and Widening need reprograming when platform changes
  - E.g. int8 feature map/weight → accumulator widened to int32



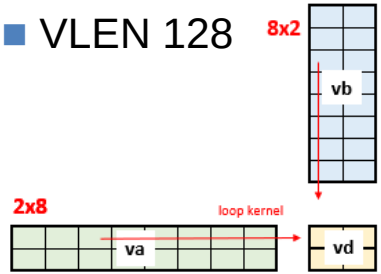
SW Scalability Innovations

- Achieve SW Portability by incorporating following innovations
  - ① Scalability management for vector registers
  - ② Diverse tile shape support using hybrid inner/outer products
  - ③ Flexible matrix multiplication unit for versatile programs
  - ④ VLEN 2048 → 4bits imm fields
- amm vd, vb, va, imm
  - vd = va.px/py * vb
  - portions managed by imm argument

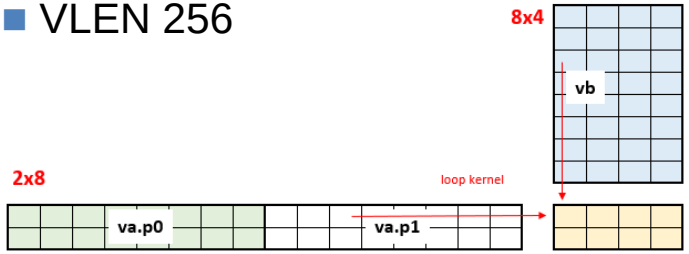


Scalable Programming Model (E.g int8 widen)

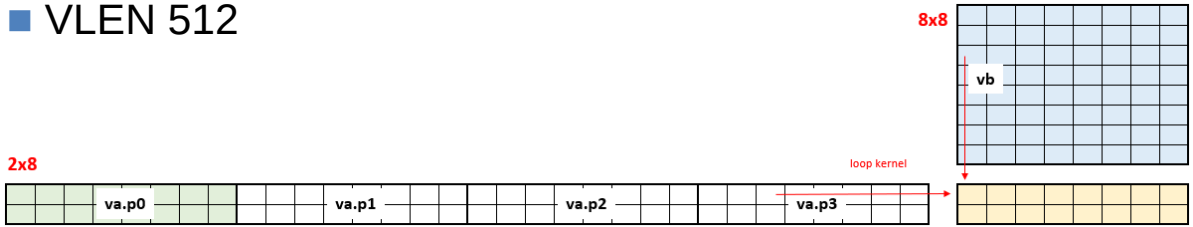
- VLEN 128



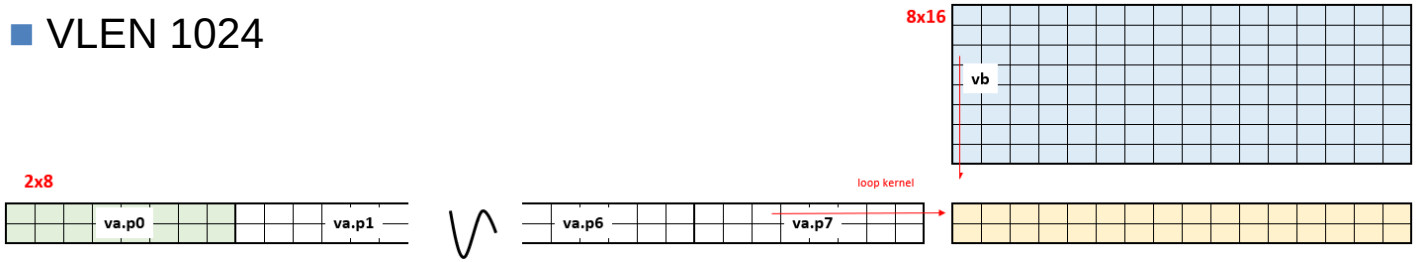
- VLEN 256



- VLEN 512



- VLEN 1024



Scalable Programming Model

```
SEGS = VLEN >> exp(seg_size);
void gemm_kernel() {
…
 vector<int8> va,vb;
 vector<int32> vc;
 while(k>0){
  vload va,[mem_a];
  for(i=0;i<SEGS;i++){
    vload vb,[mem_b];
    amm vc, vb, va, i;
  }
  k-=vl;
 }
…
}
```

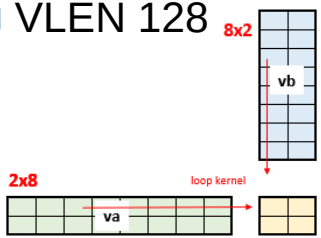
Source level scalability
- no need to rewrite program

int8 widen to int32, or int32 to int32

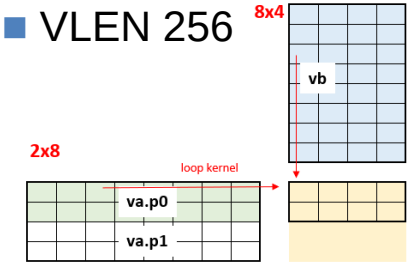
- Support binary portable if segs loops kept with gpr i

Optimal Programming Model (E.g. int8 widen)

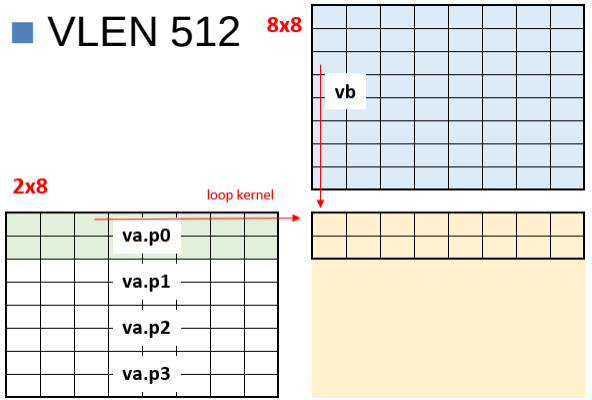
- VLEN 128



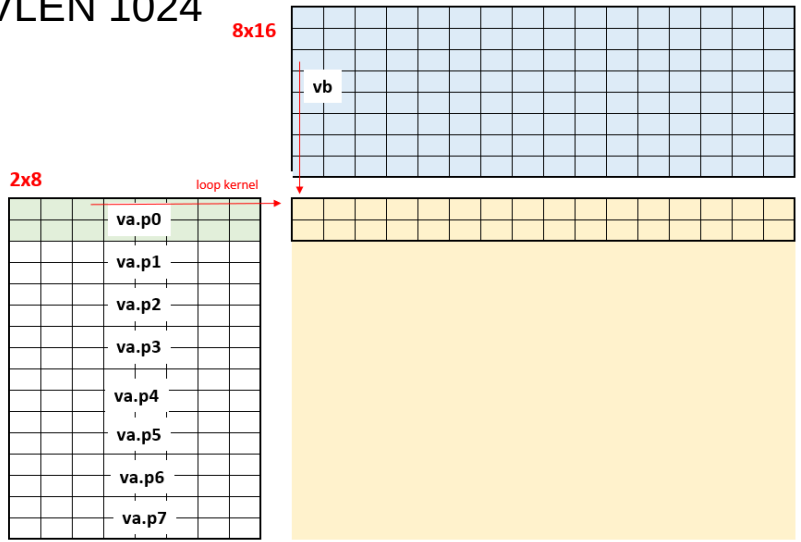
- VLEN 256



- VLEN 512



- VLEN 1024



Optimal Programming Model

```
SEGS = VLEN >> exp(seg_size);
void gemm_kernel() {
…
 vector<int32> va,vb;
 vector<int32> vd[SEGS];
 while(k>0){
  vload va,[mem_a];
  vload vb,[mem_b];
  for(i=0;i<SEGS;i++){
    amm vc, vb, va, i;
  }
  k-=vl;
 }
…
}
```

int8 widen to int32, or int32 to int32

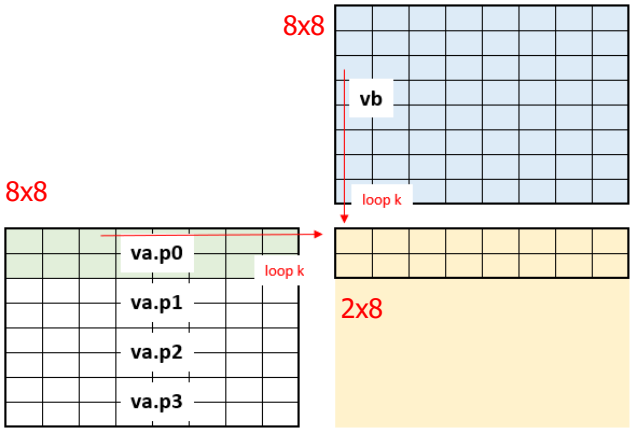
- Support binary portable if segs loops kept with gpr i

Widen Support

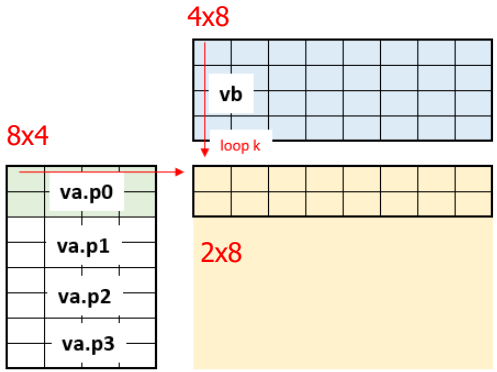
Widen	Example Scenarios	Notes
4x	Int8 → Int32	
2x	Int16 → Int32, FP16 → FP32	
1x	Int32 → Int32, FP32 → FP32, FP16(BF16)→FP16(BF16)	

- Example VLEN 512

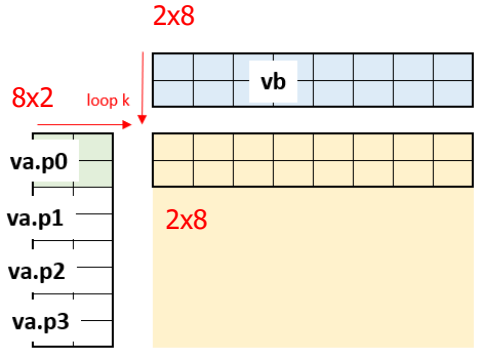
widen4x



widen2x



widen1x



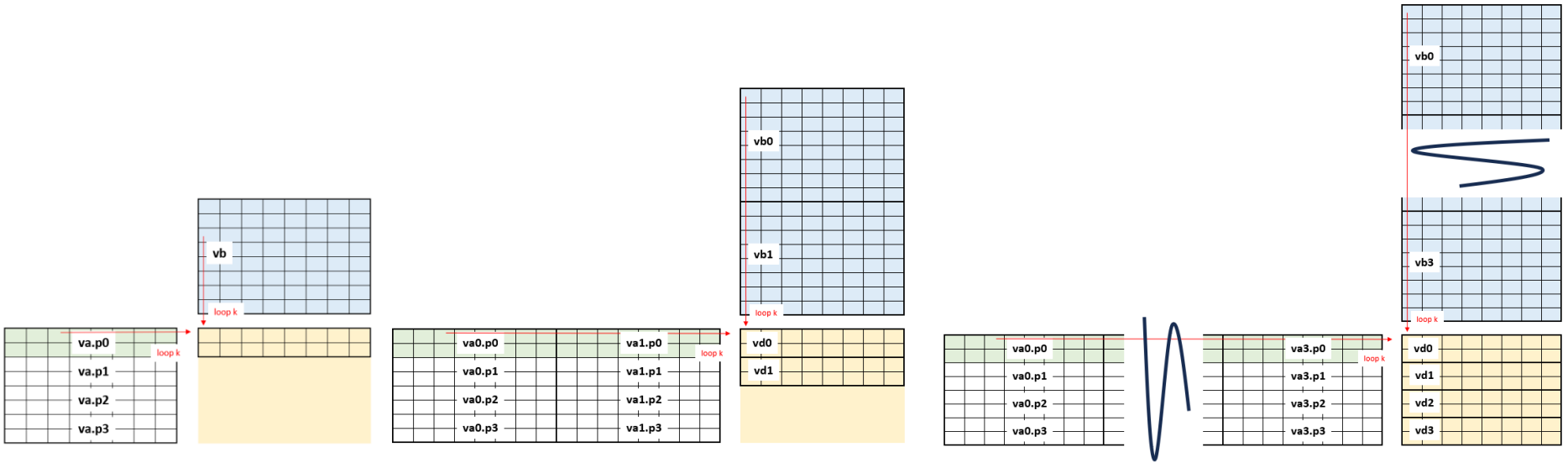
LMUL Support

- Example VLEN 512

LMUL=1

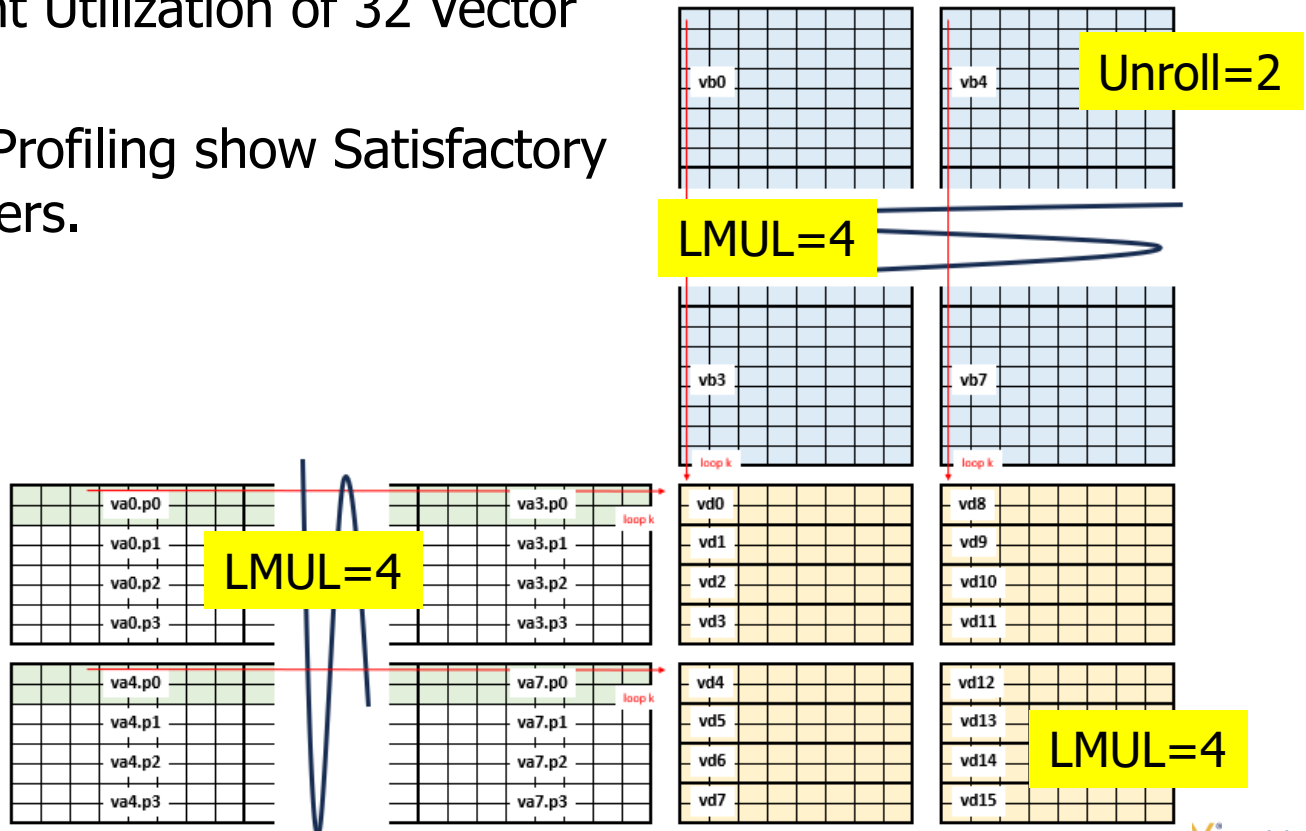
LMUL=2

LMUL=4



LMUL + Unroll Support

- Elegant and Efficient Utilization of 32 Vector Register Files.
- GeMM Experiment Profiling show Satisfactory Performance Numbers.

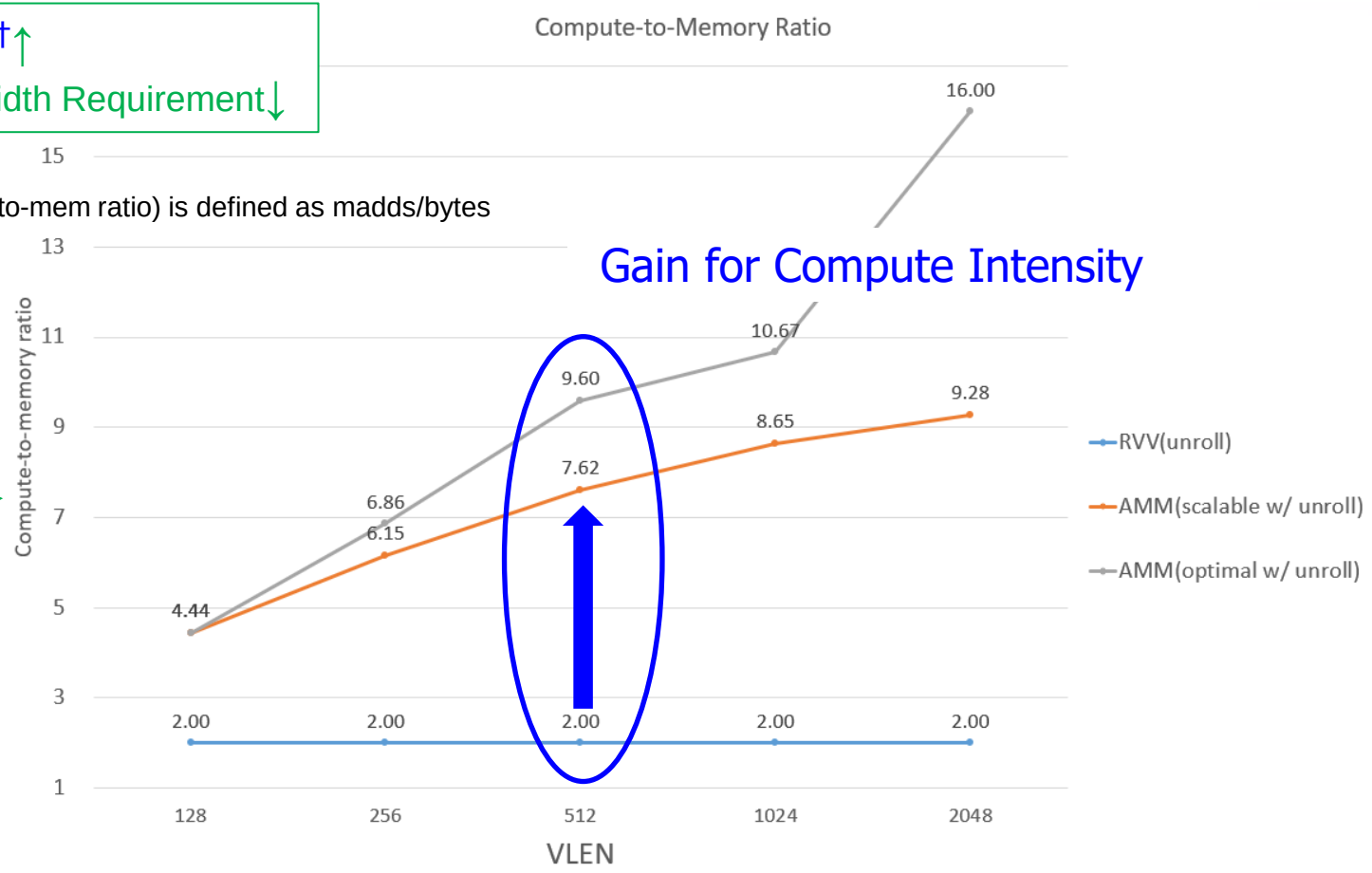


Compute Intensity η (Compute-to-Mem Ratio, int8→int32)

Compute-to-Mem Ratio†↑

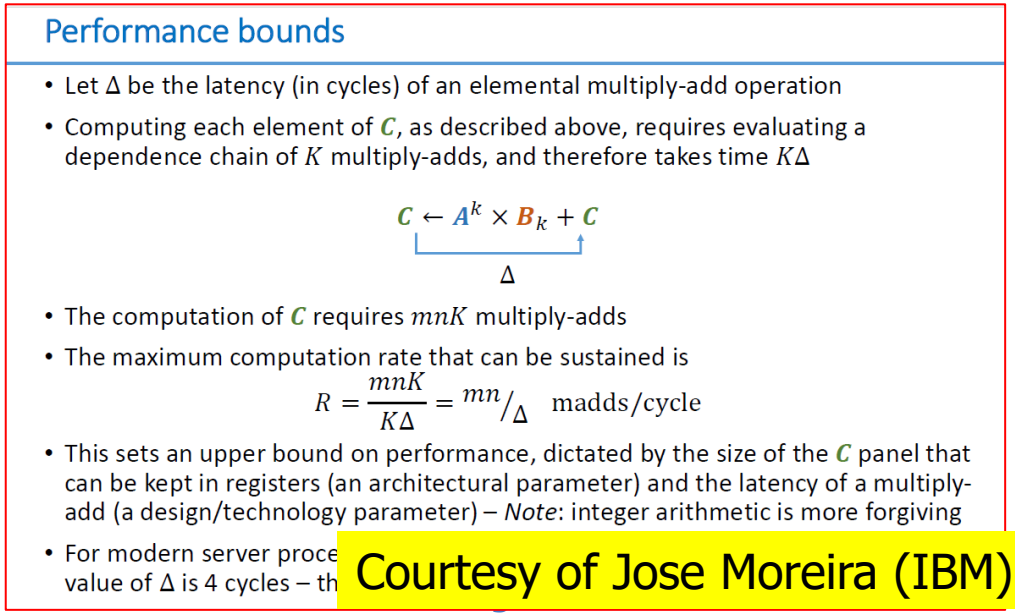
Memory Access Bandwidth Requirement↓

†:compute Intensity η (compute-to-mem ratio) is defined as madds/bytes



Max Compute Rate R (1/)

- Reference:
  - https://github.com/riscv-admin/vector/blob/main/minutes/2023/2023-10-02/Matrix_2023-OCT-02.pdf
  - Refer to Jose Moreira (IBM) presentation on 2023/10/2
- Architecture space 32 vector register as 32*L (Words)
- m', n' are register unrolling for SW Implementation freedom factor



Performance bounds

- Let Δ be the latency (in cycles) of an elemental multiply-add operation
- Computing each element of C, as described above, requires evaluating a dependence chain of K multiply-adds, and therefore takes time $K\Delta$

$$C \leftarrow A^k \times B_k + C$$

Δ

- The computation of C requires mnK multiply-adds
- The maximum computation rate that can be sustained is

$$R = \frac{mnK}{K\Delta} = {}^{mn}/_{\Delta} \quad \text{madds/cycle}$$

- This sets an upper bound on performance, dictated by the size of the C panel that can be kept in registers (an architectural parameter) and the latency of a multiply-add (a design/technology parameter) – *Note*: integer arithmetic is more forgiving
- For modern server proce
value of Δ is 4 cycles – th

Courtesy of Jose Moreira (IBM) Matrix_2023-OCT-02.pdf

Max Compute Rate R (2/)

VL	L (Words)	m	n	Δ (dependence latency)	Andes Proposal				Option A,B,C (4L)
					m' (Unroll)	n' (Unroll)	R		R
128	4	L/2	L/2	4	4	4	$16*(\frac{L^2}{16})$	16	16
256	8	L/2	L/2	4	4	3	$12*(\frac{L^2}{16})$	48	32
512	16	L/2	L/2	4	3	2	$6*(\frac{L^2}{16})$	96	64
1024	32	L/2	L/2	4	2	1	$2*(\frac{L^2}{16})$	128	128
2048	64	L/2	L/2	4	1	1	$(\frac{L^2}{16})$	256	256

Summary

No.	Metrics	Support	Comparison
1	Scalability	Yes	Source level scalability
2	Widen	Yes	1x~4x
3	Compute Intensity η	Up to 9.6	take VLEN 512, w/ Unroll
4	Max Compute Rate R	$m'*n'*(\frac{L^2}{16})$	m', n' for register unroll SW design factor
5	Matrix Pipes	2	Micro-architecture design factor (Machine Multi-Issue Capacity)
6	Format	int8/int16/int32 bf16/fp16/fp32	Andes: bf16 support by mode selection
7	LMUL	Yes	Depends on VRF pressure and VLEN Portions
8	VRF R/W Ports	Up to 3R/1W	Based on equivalent computing capability
9	Boundary Overhead	Almost Zero-Overhead	
10	Tiles Load/Store	Enhanced LSU	

Thank you

- In case if I don't have any immediate data or slides prepared for your question on-line. I would be very happy to provide them for discussion in future meetings or on the mailing-list.

Appendix