Graphics and ML SIG Meeting
Feb 3, 2022
10:05am PDT

https://github.com/riscv-admin/graphics

Only RISC-V Members May Attend

- Non-members are asked to please leave except for Joint Working Groups (JWG).
- Members share IP protection by virtue of their common membership agreement. Non-members being present jeopardizes that protection. Joint working groups (JWG) agree that any IP discussed or worked on is fully open source and unencumbered as per the policy.
- It is easy to become a member. Check out riscv.org/membership
- If you need work done between non-members or or other orgs and RISC-V, please use a joint working group (JWG).
    - used to allow non-members in SIGs but the SIGs purpose has changed.
- Please put your name and company (in parens after your name) as your zoom name. If you are an individual member just use the word “individual” instead of company name.
- Non-member guests may present to the group but should only stay for the presentation. Guests should leave for any follow on discussions.

Antitrust Policy Notice

RISC-V International meetings involve participation by industry competitors, and it is the intention of RISC-V International to conduct all its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

Examples of types of actions that are prohibited at RISC-V International meetings and in connection with RISC-V International activities are described in the RISC-V International Regulations Article 7 available here: https://riscv.org/regulations/

If you have questions about these matters, please contact your company counsel.

Collaborative & Welcoming Community

RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration. Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.

We are a transparent, collaborative community where all are welcomed, and all members are encouraged to participate. We are a continuous improvement organization. If you see something that can be improved, please tell us. help@riscv.org

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone.

https://riscv.org/community/community-code-of-conduct/

Conventions

- **For one hour meetings, please start at 5 after the start time** in order to allow people going to other meetings have time for a short break between meetings. 30 minute meetings start on time.
- Unless it is a scheduled agenda topic, we don't solve problems or detailed topics in most meetings unless specified in the agenda because we don't often have enough time to do so and it is more efficient to do so offline and/or in email. We identify items and send folks off to do the work and come back with solutions or proposals.
- If some policy, org, extension, etc. can be doing things in a better way, help us make it better. Do not change or not abide by the item unilaterally. Instead let's work together to make it better.
- Please conduct meetings that accommodates the virtual and broad geographical nature of our teams. This includes meeting times, repeating questions before you answer, at appropriate times polling attendees, guide people to interact in a way that has attendees taking turns speaking, …
- Where appropriate and possible, meeting minutes will be added as speaker notes within the slides for the Agenda

Agenda

- Welcoming new members (5 minutes)
- Dashboard (5 minutes)
- Discussion of a matrix multiply concept (35 minutes)

Welcoming new members

- Prof. Xichuan Zhou, Institute of Science on Brain Inspired Intelligence, Chongqing University
- Vaibhav Verma, University of Virginia
- Prof. Mircea Stan, University of Virginia
- Victor Moya, Semidynamics
- Steve, Google Research
- Nikos Stavropoulos, Think Silicon
- Alan Kao, Andes
- Dimitris Tsaliagkos
- Ken Dockser, Rivos

Dashboard concept (1 / 2)

Agreed with the Software HC chair (Philipp Tomsich)

1. Specific ISA instructions (Abel can do this section 1 of the dashboard)

1.1 Support for BFloat16 **(Ken Dockser, Rivos, volunteered)**
- Conversion as in TVM
- Conversion as in Pytorch/Glow
- Conversion as in TensorFlow
- Widening dot product.

1.2 Support for 8-bits and 16-bits arithmetic
- Saturating arithmetic
- Widening dot product.
- (Optional) conversion to and from quantized datatypes

1.3 Support for Matrix multiply
- Matrix product for small matrices (as in Nvidia tensor cores).
- Convolution and pooling (with pooling being a "degenerated convolution").

Dashboard concept (2 / 2)

2. Software frameworks and libraries

- TVM
- TensorFlow / TensorFlow Lite
- IREE (based on MLIR)
- (Optional) PyTorch / Glow
- (Optional) BLAS

3. Industry standard benchmark results:

- ResNet50 retired instruction count on QEMU: only with vector instructions vs. including all the proposals.
- (Optional) ResNet50 results for "comparable devices".

Matrix Multiply for RISC-V Concept

Abel Bernabeu <abel.bernabeu@esperantotech.com>

Motivation and goals

- Increase the level of **abstraction beyond dot product:**

  Once operations are lowered to dot products by a machine learning or shader compiler, the acceleration factor from the least to the most area intensive microarchitecture is between 1 and the matrix size “n”.

  However, we would like to support microarchitectures more aggressively trading area for performance acceleration factors from 1 to n^3.

  By widening the range, we make the tradeoff more of a choice by the microarchitecture designer and less of a constraint imposed by the ISA, allowing new performance peaks on key benchmarks like Resnet50 and making RISC-V future-proof

- Concepts can and must **build nicely on top of the existing Zve**

- The ISA should assume that **Zvediv** or an equivalent extension is present

Use cases on Graphics and ML

Related SPIR-V instructions for graphics and GPGPU:

- OpVectorTimesMatrix
- OpMatrixTimesVector
- OpMatrixTimesMatrix
- OpTranspose (optional for matrix multiply)

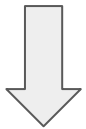
ML uses:

- Convolutions, fully connected layers, big matrix multiply

Same use cases that benefit from Nvidia’s Tensor Cores, if that is mention is allowed

Going from a “vector” to a “matrix” architecture

Vector:	Common length for sources and destination operands



Matrix:	Source operands are matrices with different sizes (although power of 2 and compatible for operation) Destination has an implicit shape, that depends on the operation and the sources

One product to rule them all

matrix x matrix
matrix x vector
vector x matrix
dot

Optional:
scalar x matrix
scalar x vector
even a transpose!

xmatmul

```
a : IxK elements of ElemTy, row or col major
b : KxJ elements of ElemTy, row or col major
c : IxJ elements of ElemTy, row major

for (i = 0; i < I; ++i) {
  for (j = 0; j < J; ++j) {
    AccTy acc = 0
    for (k = 0; k < K; ++k)
      acc += a(i,k) * b(k,j)
    c(i,j) = cast<ElemTy>(acc)
  }
}
```

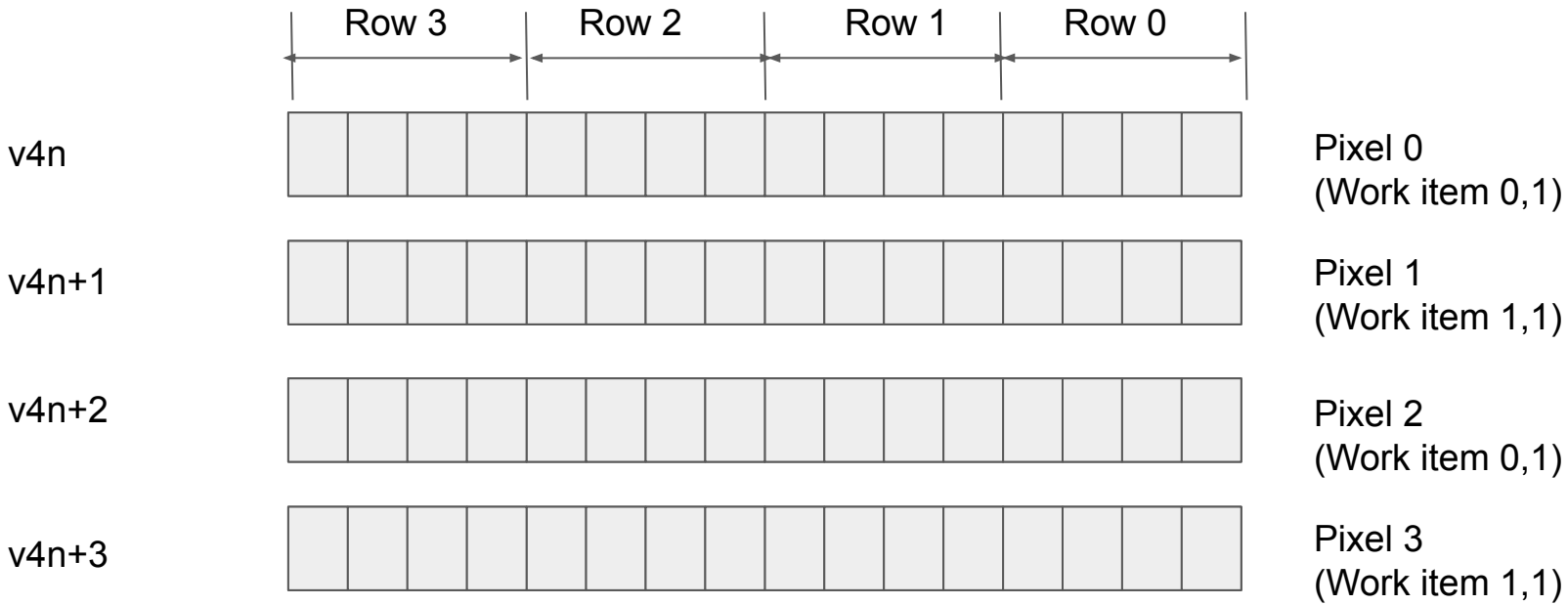
Suitable for Graphics and ML

Matrices that fit in vector registers

Matrix size at least 4x4, which is the maximum size exploitable by graphics (acceleration factors from 1 to 4^3)

Support for low precision and quantized data types as needed by ML

Example: a 4x4 row major matrix per pixel




```
2048 bits register group
vtype: VLEN=512 bits, LMUL=4 regs, SEW=32 bits
```

mtype: complementing vtype for matrices

Zve	Zve + New Zvediv + matrix multiply
`vtype`	`(vtype, mtype)`
Variable allocation	Variable allocation (vtype) Replication model across elements in workgroup (mtype) Dimensional structure (mtype)

mtype CSR layout

STYPE	Scalar type
LPWI	LHS: per work-item data?
LW	LHS: log2 width (how many bits? 2x2 minimum)
LH	LHS: log2 height
LMAJ	LHS: row or col major
LMOD	LHS: vreg, scalar splat or IDENTITY
RPWI	…
RW	…
RH	…
RMAJ	…
RMOD	...

mtype.LPWI (LHS per workitem)

Data is common for all the workitems

or

data is different for each workitem

It avoids data duplication when unneeded.

mtype.LW/LH (LHS width and height)

Width and height as log2 for compact encoding

mtype.LMAJ (LHS majorness)

Row or column major

Program flips the bit for transposing

mtype.LMOD (LHS mode)

- vreg: the content of given source vector register is interpreted as a matrix
- scalar splat: the content of one source scalar register (of X type) is replicated across all the matrix elements
- identity: source operand is ignored and the identity matrix used

An available encoding space for Graphics and ML

major opcode =1100111b (`jalr`), funct3 = 100b

22 bits per instruction

7 bits secondary opcode
5 bits destination
5 bits source 1
5 bits source 2

Questions?

Backup Slides