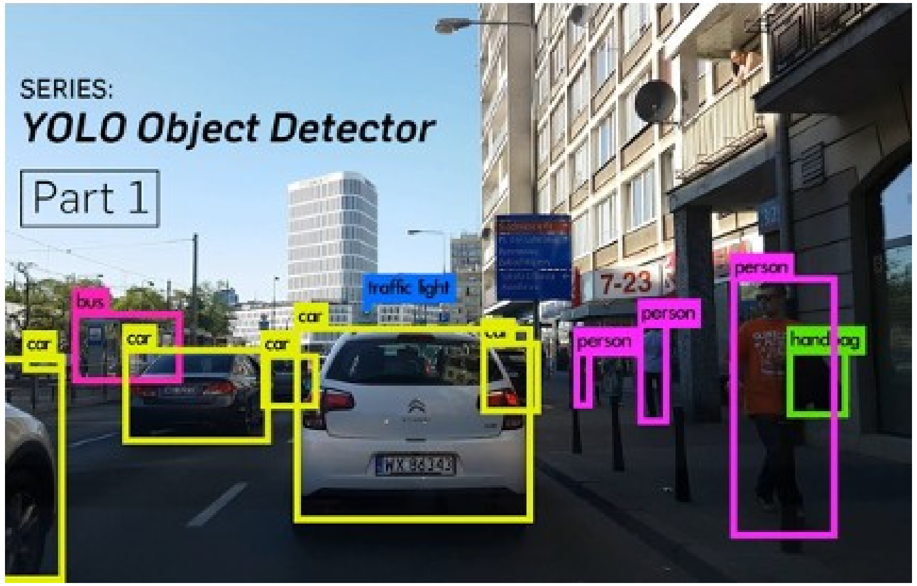
Challenges and Opportunities in the Co-design of Convolutions and RISC-V Vector Processors

Sonia Rani Gupta, Nikela Papadopoulou and Miquel Pericàs

Chalmers University of Technology, Sweden

Motivation



Inference via Convolutional Neural Network (CNNs) require high throughput and low latency

Vector processors can offer
- low latency
- high performance
- energy efficiency

Can we use long vector architectures (eg RISC-VV)?

Image credit:
https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

Background: CNN inference

YOLOv3 Object detection

- **Convolutional layer:** ~98% of total time.

Implementation for convolutional layer:

- im2col+GEMM
- **Winograd**
- FFT
- Direct

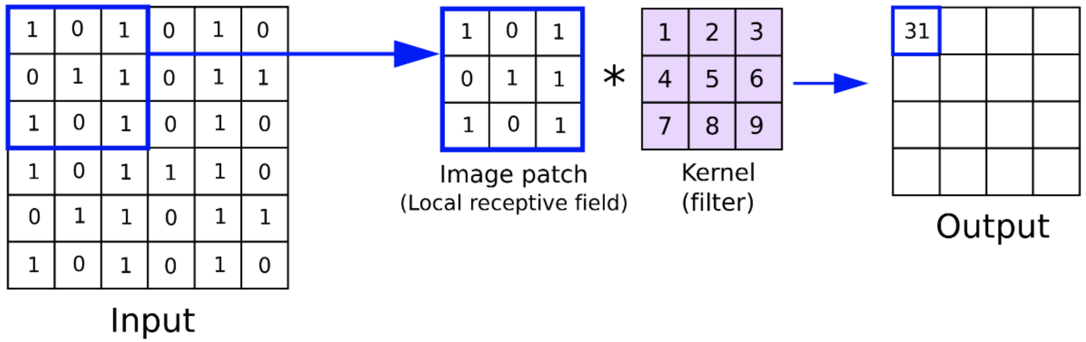


Image credit:
https://anhreynolds.com/blogs/cnn.html

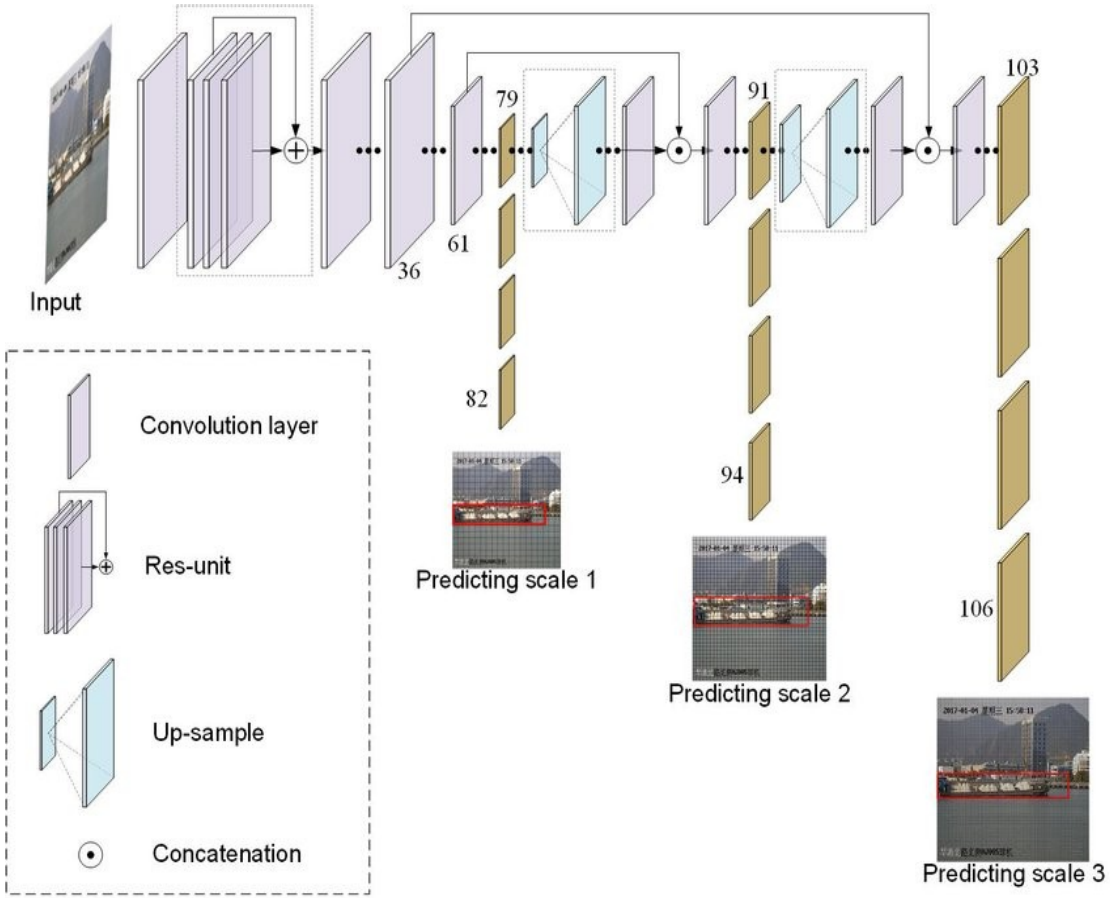


Image credit - Nie, Xin & Yang, Meifang & Liu, Wen. (2019). Deep Neural Network-Based Robust Ship Detection Under Different Weather Conditions. 10.1109/ITSC.2019.8917475.

Objective

Algorithmic Optimizations:

- Utilize the vector unit and vector registers effectively.
- Vectorize the Winograd algorithm effectively by leveraging the available EPI intrinsics

Hardware Parameters Tuning

- Vector unit: how long should vector lengths be?
- Caches: how large should caches be for different vector lengths?

Objective

Algorithmic Optimizations

Utilize the vector units and vector registers effectively

Hardware Parameters

Tune vector units, caches, and on-chip vector parallelism

Co-design study

Design effective vector architectures for high performance CNN inference.

Lack of dual approach has the risk of missing important insights

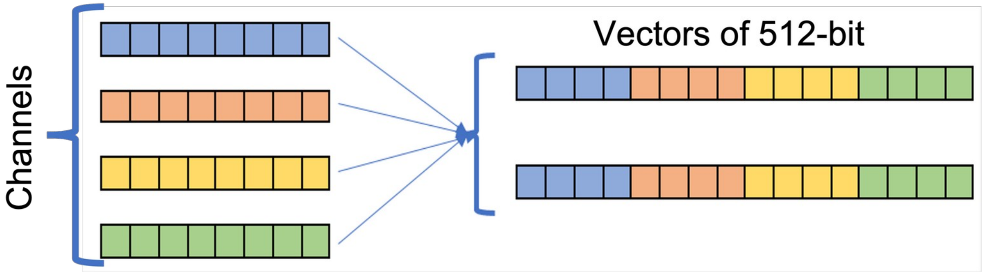
Experimental Setup

- Network models:
  - YOLOv3: 75 convolutional layers out of 107.
  - VGG16: 13 convolutional layers out of 16
  - Implemented in Darknet framework
- Algorithmic implementation
  - NNPACK library for Winograd implementation
- Hardware Exploration:
  - RISC-V Vector Extension: Gem5 Simulator*
- Compiler
  - RISC-V LLVM/Clang toolchain from the European Processor Initiative (EPI)

*Gem5 Simulator – plctlab. 2022. plct-gem5 (https://github.com/plctlab/plct-gem5/), supports v1.0 “V” extension with max VL of 4096 bits

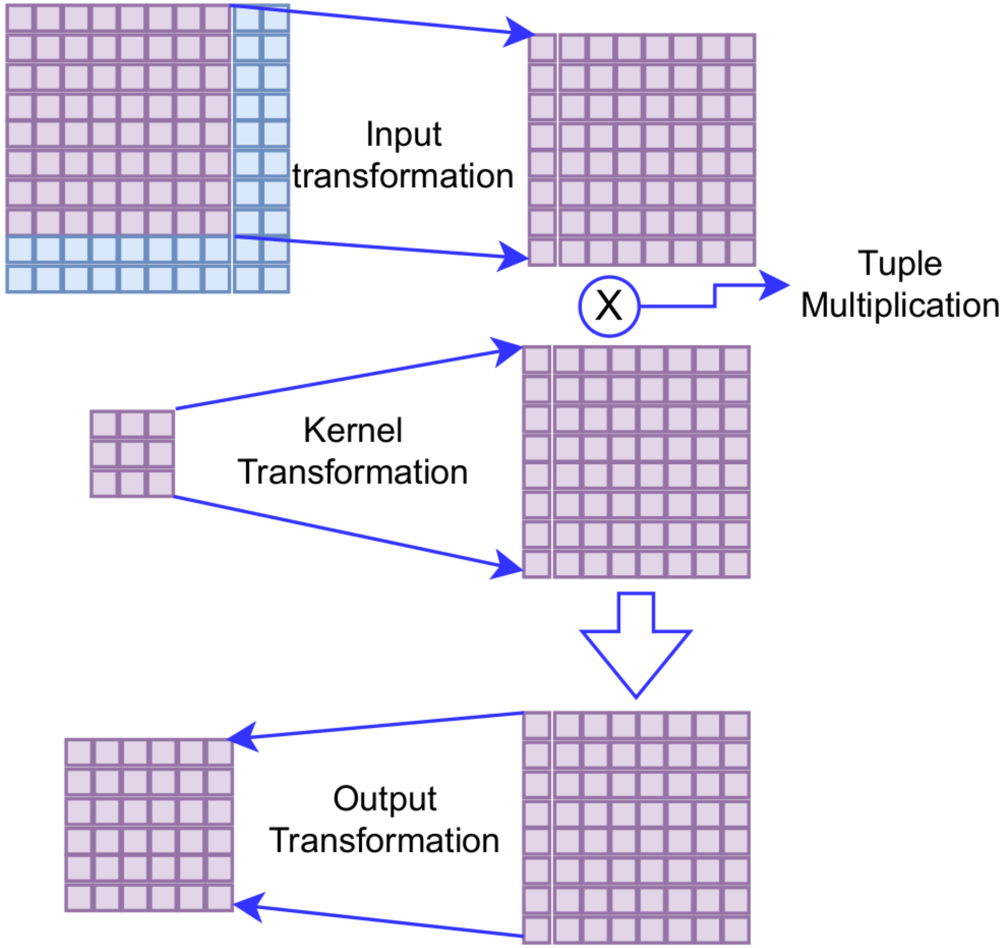
Winograd: Algorithmic Optimizations

Transformations:
- 8x8 tile from one channel (NNPACK)
- Inter-tile Parallelism across the channels**
- Similarly, 32 channels to utilize 4096-bit VL



1 row of 8x8 tile from 4 channels

Tuple multiplication
- Increase tuple size from 3 to 32 with 4 elements in each block to utilize longer vector length.

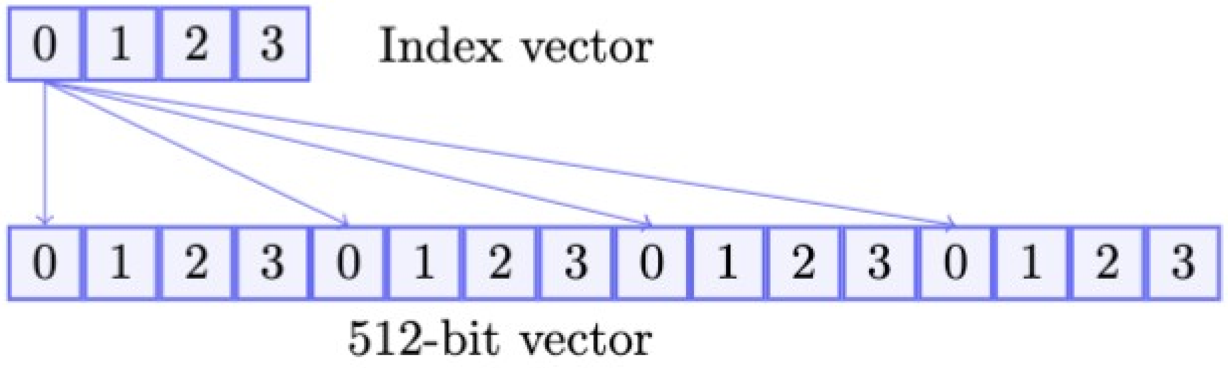


F(6x6,3x3) -> m+r-1 x m+r-1 tile
[m= output, r =kernel]
6x6 output and 3x3 kernel size = 8x8 Tile

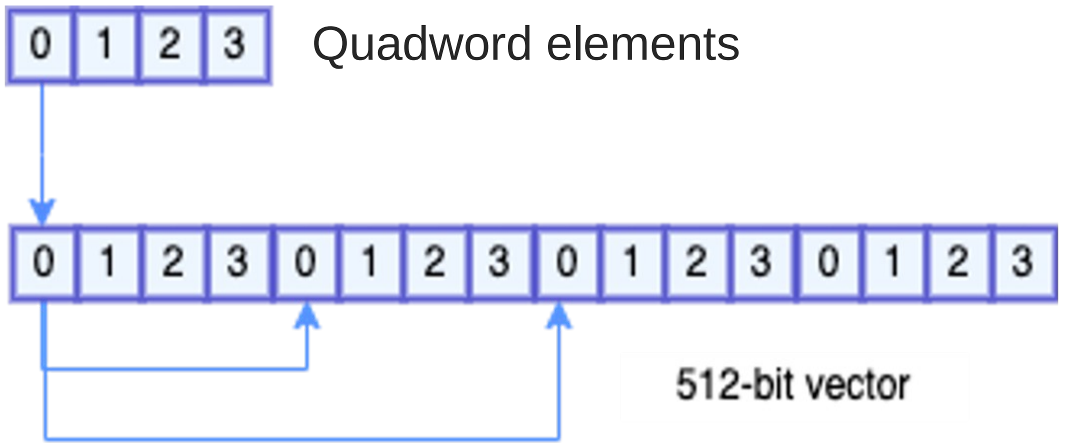
**Sonia Rani Gupta, Nikela Papadopoulou, and Miquel Pericas. 2023. Accelerating CNN inference on long vector architectures via co-design. In 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, 145–155.

Challenge #1: Tuple Multiplication

- Operation: Load Quadword elements in a vector and replicate:
  - No specialized RISC-VV Instruction
- We test two alternatives
  - **Implementation 1: Indexed Load**
  - **Implementation 2: Slideup instructions**



Implementation 1



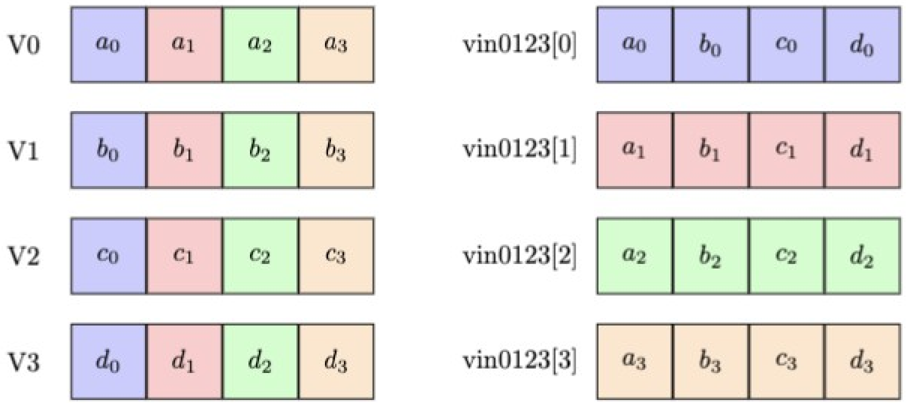
Implementation 2

Implementation 2 with slideup is ~2.3X faster than implementation 1 with indexed load.

Having specialized instruction likely to be faster, and reduce register pressure.

Challenge #2: Transformations – Transpose four vectors

- **Operation:** Transpose of 4 vectors in all transformations
  - Again, no RISC-VV instruction is available.
  - **EPI custom extension provides transpose with 2 vectors.**
  - **We tested two alternatives:**
    - **Implementation 1**: unit-strided store followed by Indexed load
    - **Implementation 2:** Strided store followed by unit-strided load



Example for transposing 4 vector registers having elements from 1 channel

No significant difference in performance with both implementations

Potential RISC-VV extension: vector transpose of 4 vectors, eliminates need for extra memory operations

Challenge #3: Transformations - Calling Conventions

Problem: Cannot pass references to vector registers as parameters to a procedure

- require intermediate vector registers to store the intermediate vector data.
- ~30 lines of code at 6 places in the input transformation kernel. Problems:
  - Register spilling
  - Less Programmability
- **Potential Workaround: Macros** can improve programmability, but it will still be required to have **intermediate registers. Problem of register spilling will remain***

Being able to pass references to vector registers would improve programmability and reduces the chances of register spilling

*As the extended need for intermediate registers can still cause register spilling

VGG16: Analysis

VGG16:

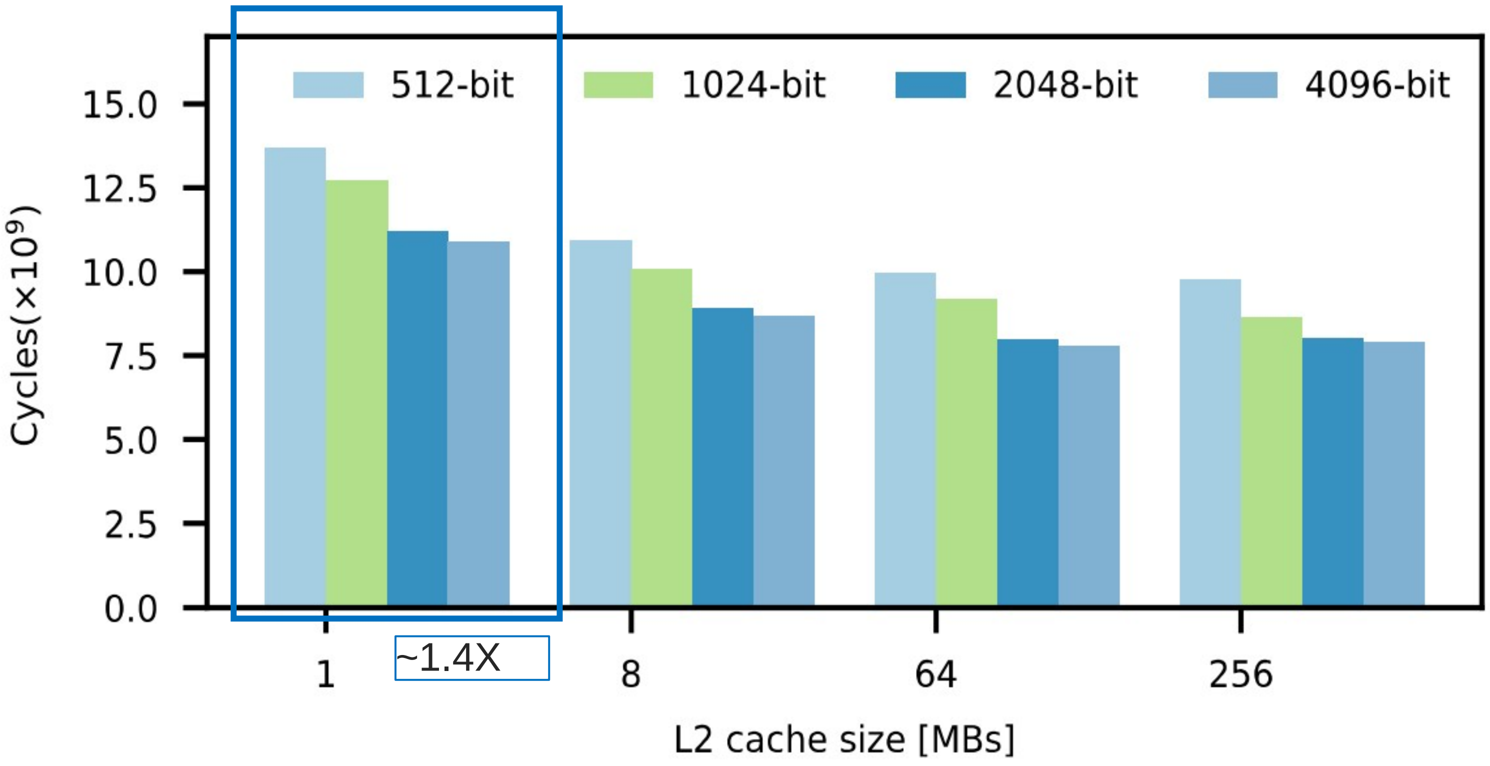
- 3x3 kernel size with stride 1: **Winograd**
- All the layers use Winograd algorithmic optimizations

Comparison with im2col+GEMM:

- 2048 bits VL and an L2 cache of 1MB modeled with gem5
- **1.2x performance improvement** Compared to the **pure im2col+GEMM** approach.
- Similar performance compared to our optimized ARM-SVE implementation (on gem5)

HW Design Space: VGG16

Impact of vector lengths and L2 cache size with Winograd on RISC-VV@gem5 for VGG16.



Impact of Vector lengths:

- **No scalability beyond 2048-bit.**
- No significant difference in the number of instructions from 2048-bit to 4096-bit vector lengths

Impact of L2 caches from 1MB to 64MB:

- ~1.3X performance improvement

No performance improvement beyond 64MB L2 cache

Our Winograd implementation does not have a high cache requirement. 2K vector length with 64MB caches can provide up to ~1.8x speedup

YOLOv3: Analysis

CHALMERS
UNIVERSITY OF TECHNOLOGY

YOLOv3: Hybrid approach

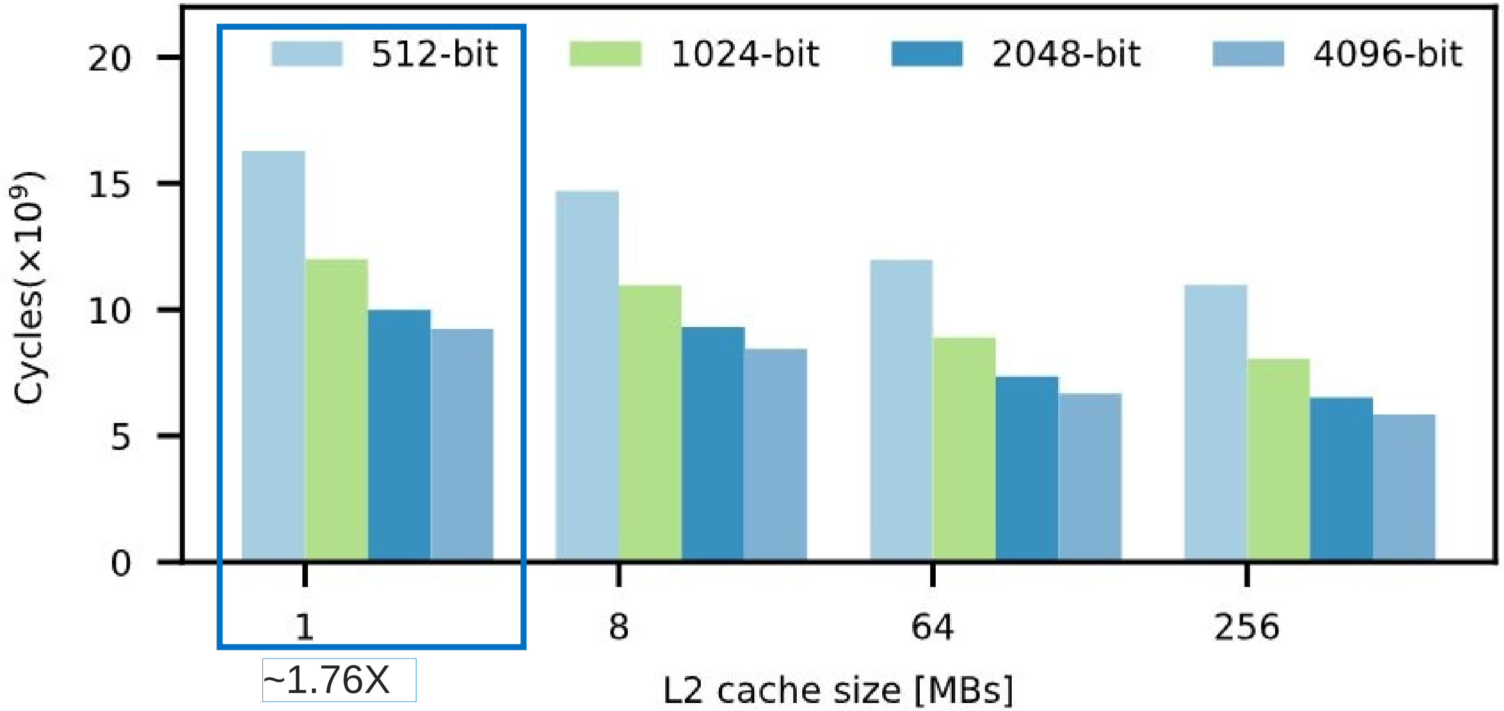
- 1x1 kernel size: **im2col+gemm**
- 3x3 kernel size with stride 1: **Winograd**
- 3x3 kernel size with stride 2: **im2col+gemm**
- Only 5 layers use Winograd out of 20 layers.

Comparison with im2col+GEMM:

- First 20 layers with 2048 bits VL and an L2 cache of 1MB modeled with gem5
- **8% performance improvement** compared to the **pure im2col+GEMM** approach.
- Similar performance compared to our optimized ARM-SVE implementation (on gem5)

HW Design Space: YOLOv3 Hybrid

Impact of vector lengths and L2 cache size with Winograd on RISC-VV@gem5 for YOLOv3 (20 Layers)



Impact of L2 caches from 1MB to 256MB:

- Upto 1024-bit: **1.5X**
- Beyond 2048-bit: **~1.6X**

4K vector length with 256MB can provide up to ~2.6x speedup. This is mainly due to im2col+GEMM scaling

Discussion on tools

Gem5@ RISC-V (https://github.com/plctlab/plct-gem5/): Tightly Integrated VPU

- Supports v1.0 Vector extension
- Very long vector lengths beyond 4096-bit are not supported yet.
- No out of order model or prefetching support
- Models a constant latency for all the vector instructions. In practice, the latency of the instructions will depend on the implementation of RISCV-V.

****Gem5@RISC-V:** Decoupled VPU with maximum of 8 vector lanes.

- No Prefetching support and no out of order model
- Supports 16384bit VL
- No longer maintained
- Supported 0.7 RISC-V Vector extension

SPIKE:

- Emulator with 4096-bit VL (used mainly for validation in our work)

**C. Ramírez, “A risc-v simulator and benchmark suite for designing and evaluating vector architectures,” ACM Trans. Archit. Code Optim., vol. 17, no. 4, Nov. 2020. [Online]

Conclusion

Goal: Design Space Exploration of RISC-VV by studying combined implications of algorithmic optimizations and HW parameters tuning with Winograd algorithm for CNN

Conclusions:

- We identified several potential extensions to RISC-VV: LoadQuadword+replicate, Transpose of four vectors, and passing references to vector registers.
- We implemented alternatives for this limitations. Final performance was similar to ARM-SVE, demonstrating the performance of the proposed workarounds.
- Hardware DSE on top of optimized kernels: ~2.6X speedup for YOLOv3 and 1.8X for VGG16
- Winograd implementation scales up to 2K VL and 64MB of L2 cache. On the other hand im2col+GEMM has higher memory requirements, but also scales to longer VL.
- **Future Work**: extend the study to compare with Long Vector Direct Convolutions

Thank you