codeplay

Enable AI & HPC to be Open, Safe and Accessible to All

Enabling OpenCL and SYCL for RISC-V processors

Colin Davidson and Aidan Dodds

IWOCL 2021 : April

Outline

- Who is Codeplay and what do we do
- What RISC-V is?
- Codeplay's Software stack
- Technical challenges for RISC-V
- Our design
- Vectorization and RVV
- Next steps

Products

Acoran

Integrates all the industry standard technologies needed to support a very wide range of AI and HPC

ComputeCpp™

C++ platform via the SYCL™ open standard, enabling vision & machine learning e.g. TensorFlow™

ComputeAorta™

The heart of Codeplay's compute technology enabling OpenCL™, SPIR-V™, HSA™ and Vulkan™

Markets

High Performance Compute (HPC)
Automotive ADAS, IoT, Cloud Compute
Smartphones & Tablets
Medical & Industrial

Technologies: Artificial Intelligence
Vision Processing
Machine Learning
Big Data Compute

Company

Leaders in enabling high-performance software solutions for new AI processing systems

Enabling the toughest processors with tools and middleware based on open standards

Established 2002 in Scotland with ~80 employees

Partners

BROADCOM. CEVA Imagination
intel KMC Kyoto Microcomputer Co., Ltd. NSI-TEXE
BERKELEY LAB RENESAS SYNOPSYS
Argonne NATIONAL LABORATORY

And many more!

RISC-V

- What is RISC-V?
  - FOSS RISC Instruction Set Architecture
  - No license required
  - Simple base ISA (~40 instructions)
  - Multiple optional extensions

- Flexible Systems
  - Hard IP Blocks
  - Multiple RISC-V Cores

https://riscv.org/

The RISC-V Ecosystem

- RISC-V's place in the ecosystem
  - Growing ecosystem around RISC-V from different vendors
  - More single ISA solutions e.g. RISC-V host, RISC-V accelerator
  - Often used with bespoke components e.g. Convolution hardware
- RISC-V is growing rapidly
  - Over 1000 RISC-V members
  - Multiple large companies working with it
  - Likely to be used for a lot of new growth areas such as AI

Why OpenCL and SYCL on RISC-V

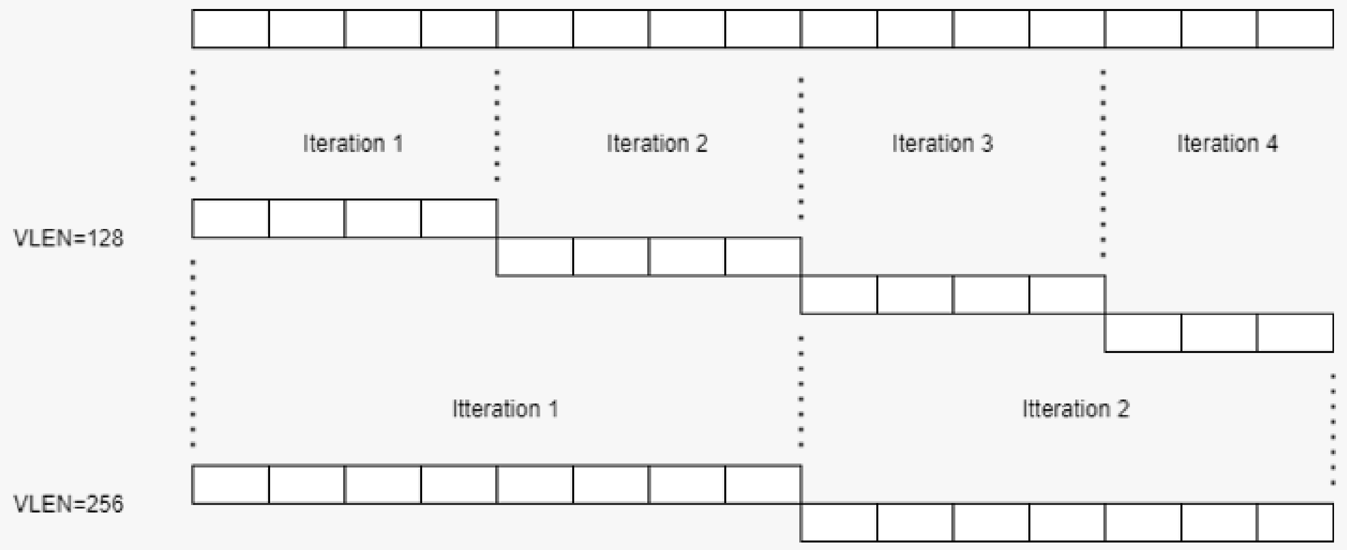
- Enable portable compute for the RISC-V ecosystem
  - Potential lots of accelerators based off the same ISA
  - Leads to maturity of product through open source community
- Scales up to high-performance targets
  - Expose and take advantage of processor extensions
  - Vector extension is important for high-performance compute
- Can port existing (compute-based) software
  - TensorFlow, SYCL-BLAS, SYCL-DNN
  - OpenCV, Halide
  - Automotive, AI

RISC-V Vector Extension

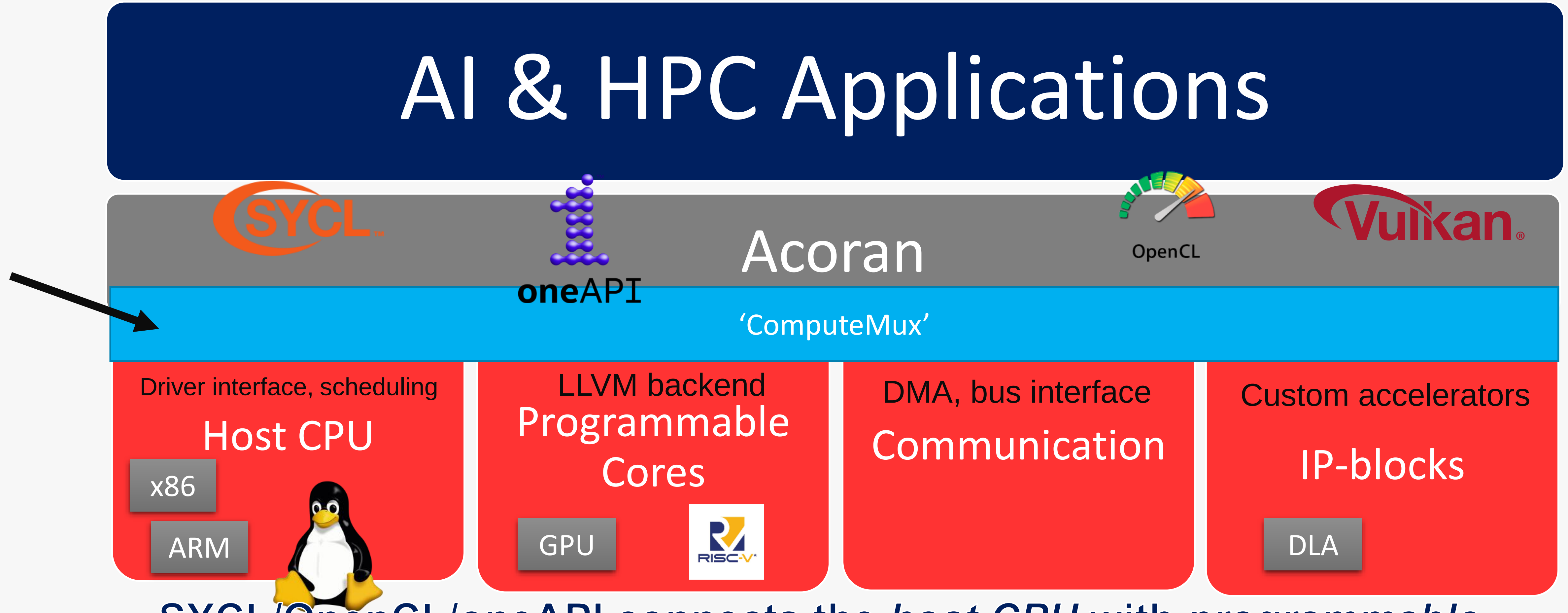
- Introduces vector support for different datatypes
  - half, float, double, etc
- Vector width operations configurable at runtime
  - The RISC-V V-extension uses scalable vectors
    - Not int2 or int4 but more like intN
  - Vector instructions will act on chunks within 'N'
    - Up to the maximum hardware-supported vector length (VLEN bits)
  - Config instructions dictate how vector registers will be used
    - e.g. element size, total required
    - returns active vector length used in following instructions

RISC-V Vector Example

- Example float operations for 15 items for different VLEN
- Configure for 15 total elements, 32 bits element size



Accelerated AI & HPC Applications



SYCL/OpenCL/oneAPI connects the *host CPU* with *programmable cores* and *custom accelerators* via *communication systems*

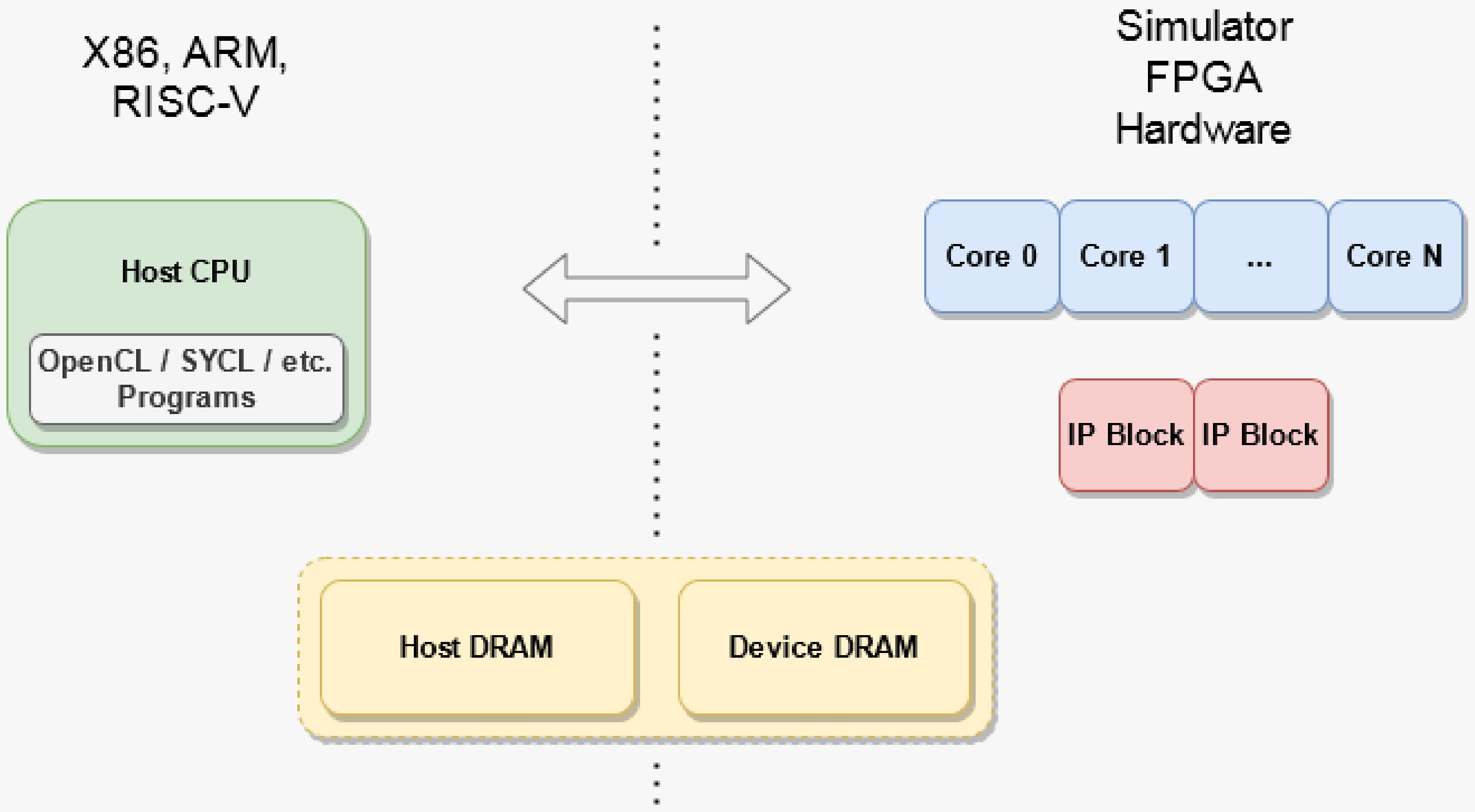
What is ComputeMux

- Interfaces our software stack to a target
  - Target is typically an accelerator e.g. GPU
  - Extensive set of entry points for complete flexibility
  - Avoid the many-to-many problem
    - (Multiply APIs * Multiple Targets)
- Exposes target specific operations
  - Communication with the target
  - Memory access
  - Kernel compilation / Loading
  - Command Queues

ComputeMux for RISC-V

- Typically tailored towards specific system
    - Typically, a single interface to hardware or simulator
    - System is mostly designed locked
- RISC-V requires a change of focus
    - Implemented during design phase of a system
    - Lots of commonality
    - Many different systems
    - Rapid development with reduce duplication

Example System

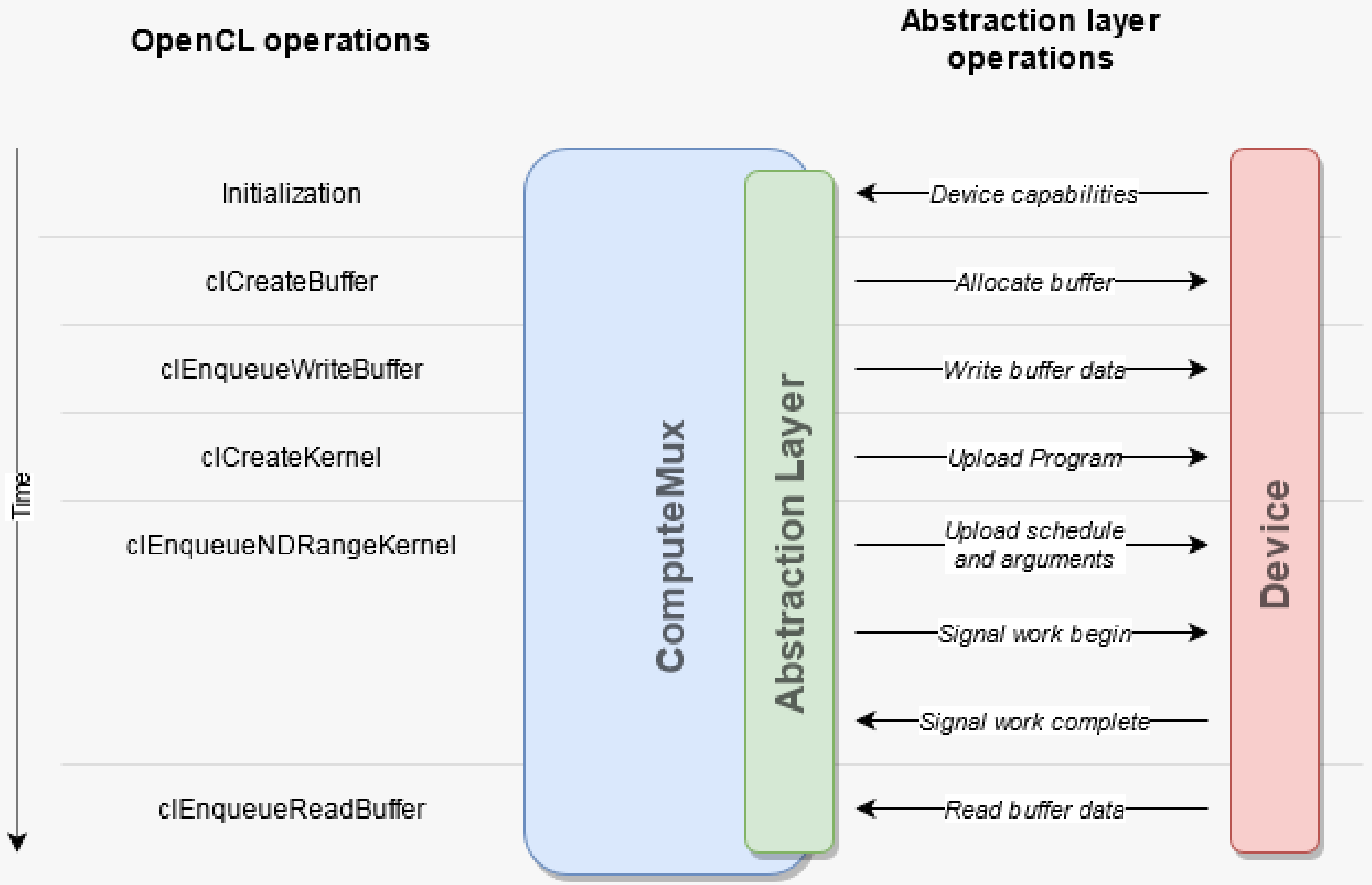


Technical Challenges

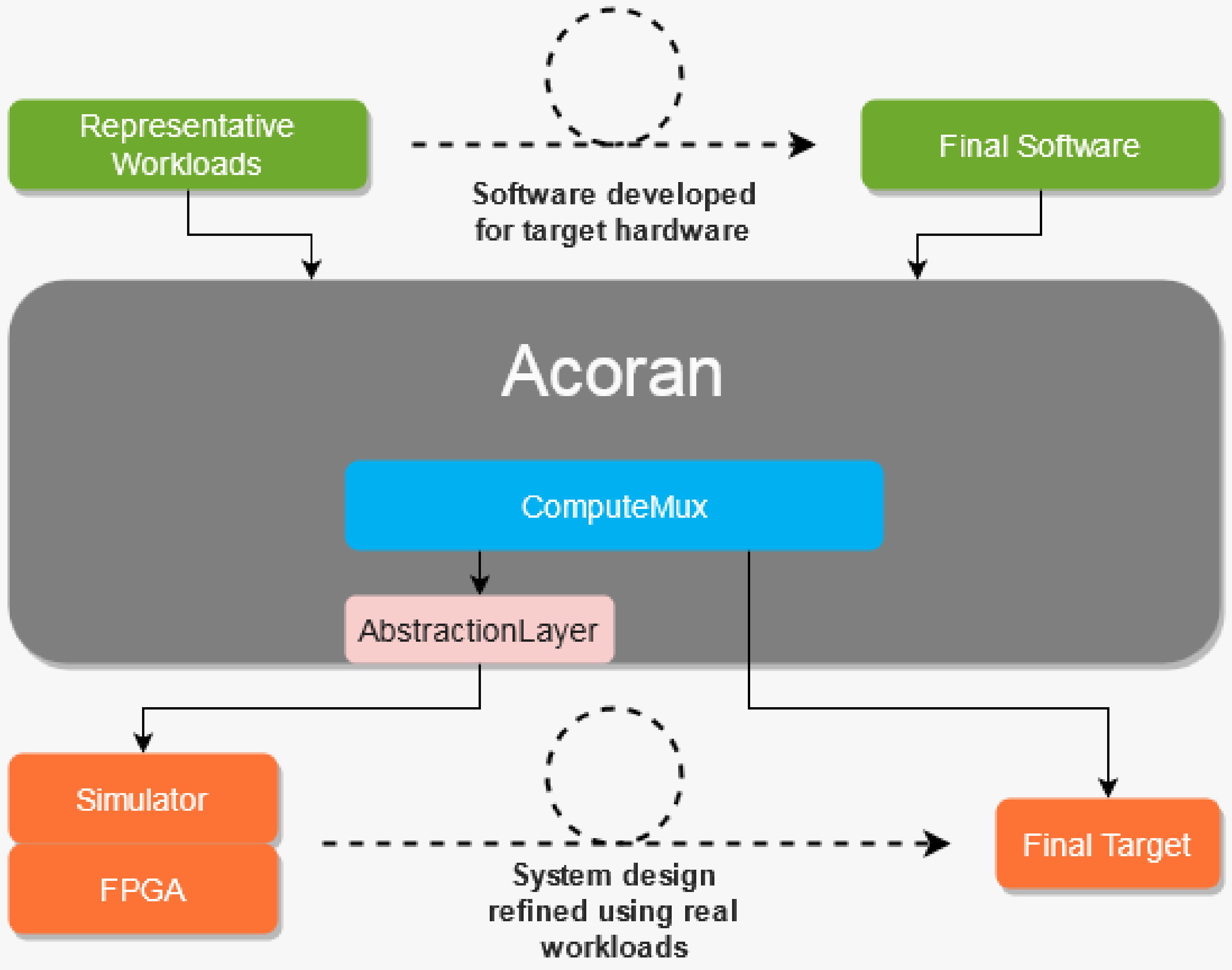
- RISC-V Specifications
    - Being defined as we develop
        - RVV vector extension
- Highly configurable per target
    - Multiple extensions (some custom)
    - Multiple simulators and hardware from different vendors
    - Different system configurations
- RVV and scalable vectors

Initial Design

- Single ComputeMux for all RISC-V targets
- Create small abstraction layer to hide target specifics
  - Focus is to hide interface details
  - Minimum set of features which OpenCL can be built upon
    - Buffer management
    - Kernel management
  - Quickly support new targets
  - Supports different configurations
- Can step beyond this layer


OpenCL operations
Abstraction layer operations
Time
Initialization
clCreateBuffer
clEnqueueWriteBuffer
clCreateKernel
clEnqueueNDRangeKernel
clEnqueueReadBuffer
ComputeMux
Abstraction Layer
Device capabilities
Allocate buffer
Write buffer data
Upload Program
Upload schedule and arguments
Signal work begin
Signal work complete
Read buffer data
Device

Supporting A New Target

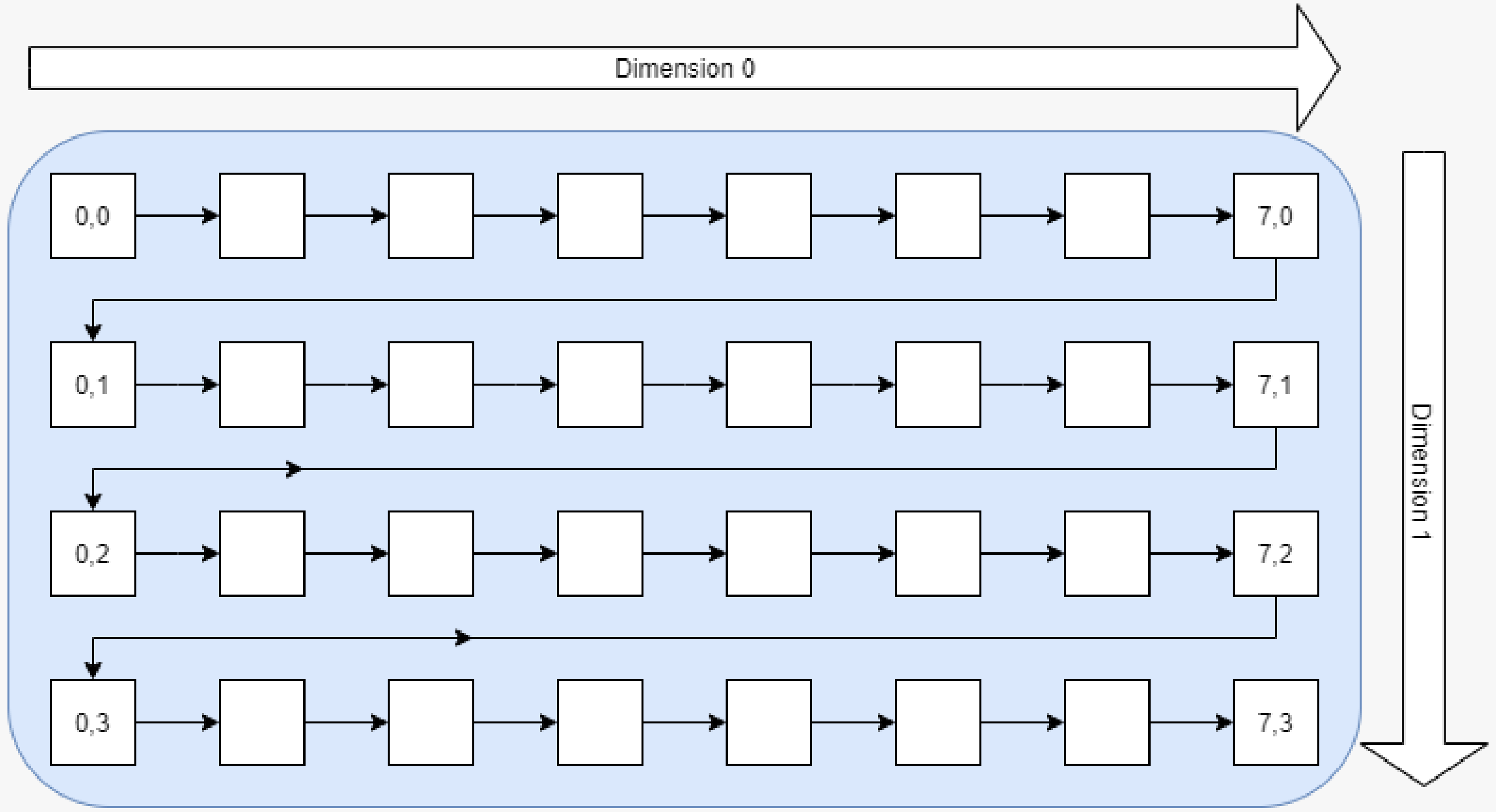


Vectorizer

- Execution of workgroups without vectorization
- Execution with whole function vectorization
- Our current vectorizer
- Challenges for the vector extension
- Design

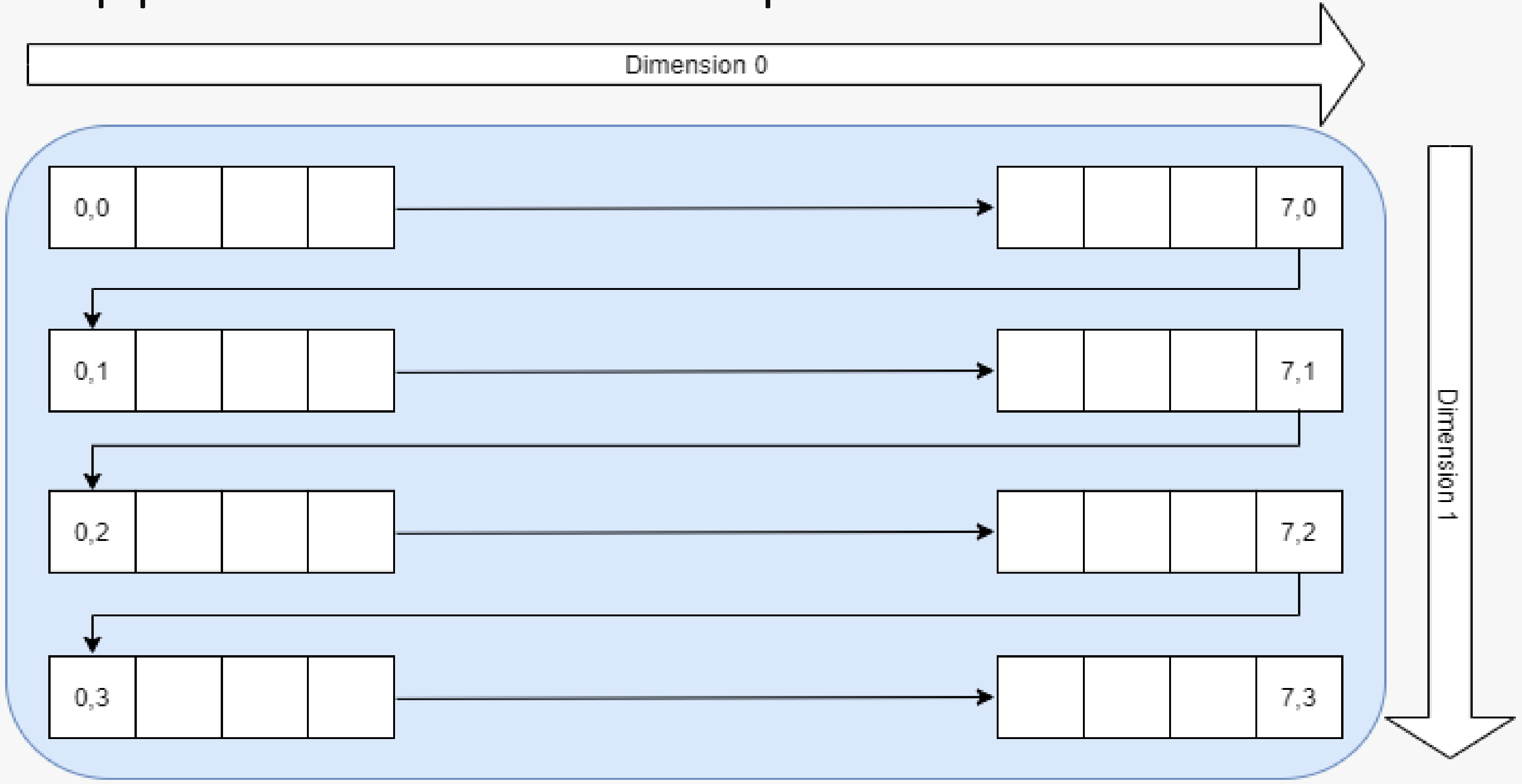
Executing Workgroups

- Executing workgroups on RISC-V can be done by looping around work items sequentially:



Vectorizing Across Workgroups

- By vectorizing across work items in a dimension we can use vector support to act on multiple work items at once:



Our Whole-function Vectorizer

- Vectorize whole kernels across work items
  - e.g. does 4 work items at a time across x dimension
  - All parts of the kernel are vectorized
- Transparent to kernel programmer
- Currently geared towards fixed-width SIMD (e.g. SSE/NEON)
- Vectorization width chosen at compile time

Vectorization Technical Challenges

- Vectorization
  - Optimal width often can't be known at compile time
  - Chosen width may not fit final work group size
    - May require vector and scalar loops to match different sizes
  - Want to support scalable vectors directly
    - More compact, efficient code
  - Vectorizer and maths library were designed for fixed-width vector support

RVV Extension Design

- Enable fixed-width vectors
  - Good for vectors in kernels
  - Helps whole function vectorization
  - Codeplay have worked with upstream to extend LLVM backend
- Improve vectorizer for RISC-V
  - Extending to produce fully scalable vectors natively
  - Performance without hand tuning or extensions
- Maths library
  - Extend to better use scalable vectors

Next Steps

- Support additional hardware
  - Bespoke IP blocks
    - AI and image-specific hardware blocks
    - DMA
- Mapping OpenCL extensions to hardware
- Support new or custom RISC-V extensions
- Continue work on scalable vectors
- Tune for performance

Summary

- RISC-V is an exciting new growth area, very relevant to OpenCL and SYCL
  - The RVV extension is particularly relevant
- Codeplay will support multiple RISC-V targets
  - Make this as simple as possible
  - Extendable through abstraction layers
  - Vector extension is a priority

We're Hiring!
codeplay.com/careers/

codeplay®
Enable AI & HPC to be Open, Safe and Accessible to All

Thank you!

@codeplaysoft

info@codeplay.com

codeplay.com