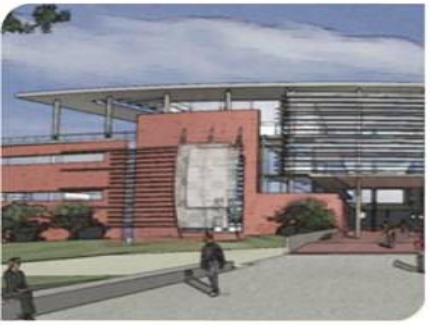
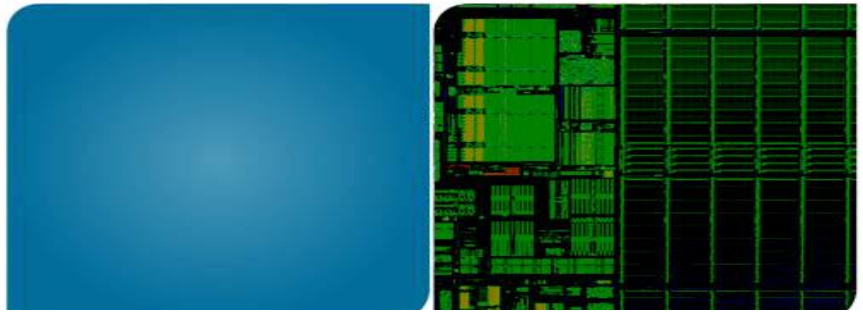
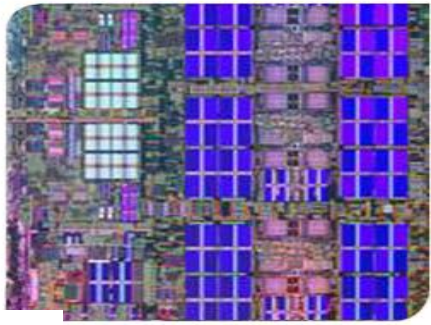
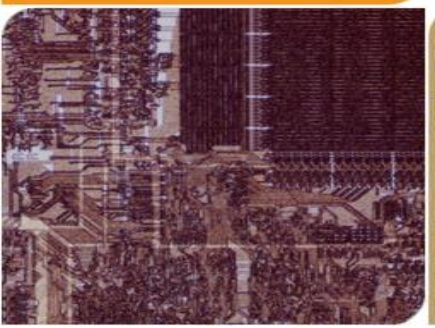




Bringing OpenCL to Commodity RISC-V CPUs

Blaise Tine, Seyong Lee, Will Gulian, Jeff Vetter, Hyesoon Kim



Georgia Tech comparch



Motivation

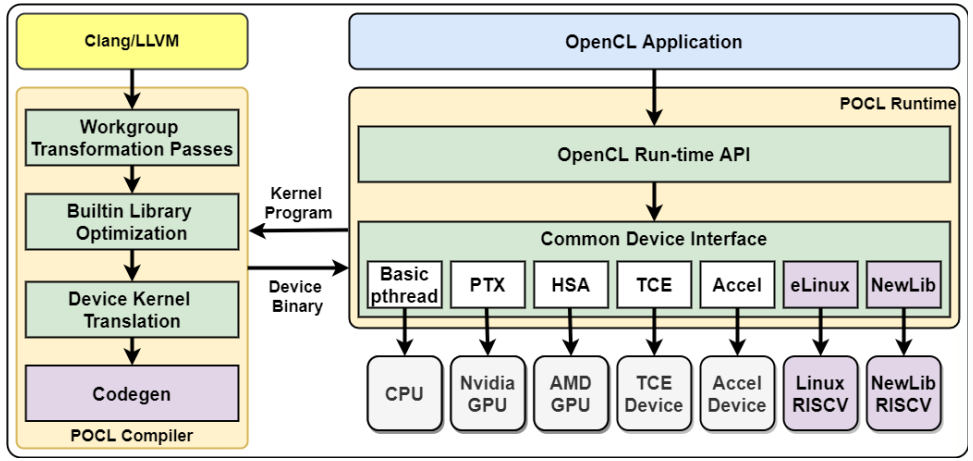
- OpenCL has proven to be effective at leveraging the parallelism in commodity multi-core CPUs and GPUs
- There is a rich variety of OpenCL applications for scientific computation
- OpenCL provides access to a unique class of benchmarks for architecture research
- There is currently no publicly available implementation of OpenCL targeting commodity RISC-V processors


Challenges

- The RISC-V ISA is very flexible
- There is a large variety of commodity RISC-V CPUs
- Most RISC-V CPUs are low-profile implementations
  - Support minimal standard ISA specification
  - May not be able to run an operating system
  - May not support a filesystem
- How to compile OpenCL runtime to target RISC-V?
- How to compile the OpenCL kernel to target RISC-V?
- How to execute multiple kernel functions?

POCL Compiler and Runtime Framework

- POCL[1]: Portable Open-Source OpenCL
  - OpenCL Compiler
  - OpenCL Runtime
- Cross-platform
  - CPUs: X86, ARM
  - GPUs: AMD, NVidia
  - Custom accelerators: TCE[2]
- OpenCL Compiler
  - LLVM-based
  - Built-in Optimized Library
- OpenCL Runtime
  - Use multi-threading on CPU
  - Use kernel offloading on GPU

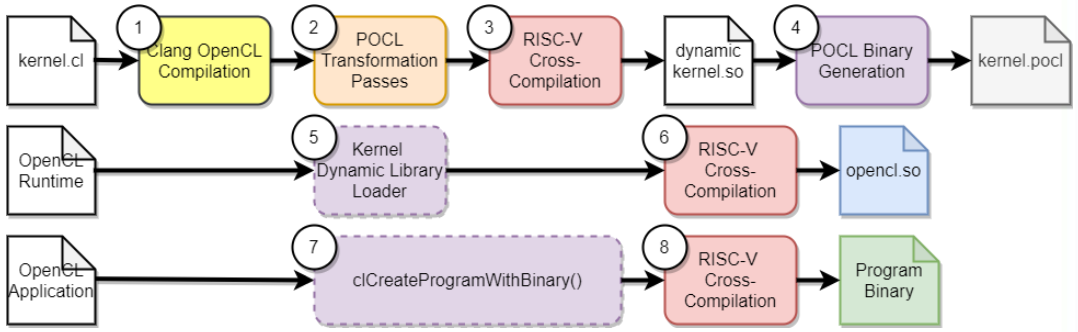


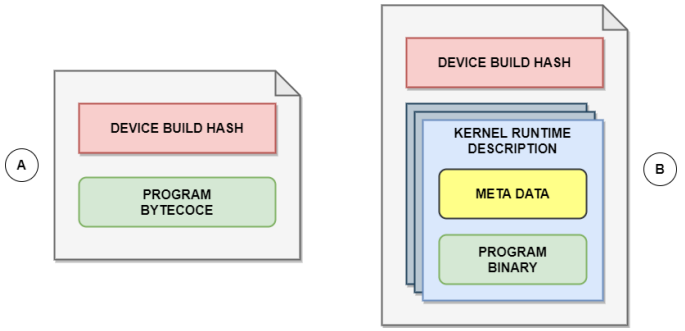
POCL RISC-V System Architecture

[1] P. Jaaskelainen et al 'pocl: A performance-portable OpenCL implementation'
[2] O. Jäskeläinen et al 'Opencl-based design methodology for application-specific processors'

OpenCL for Linux-Capable RISC-V CPUs

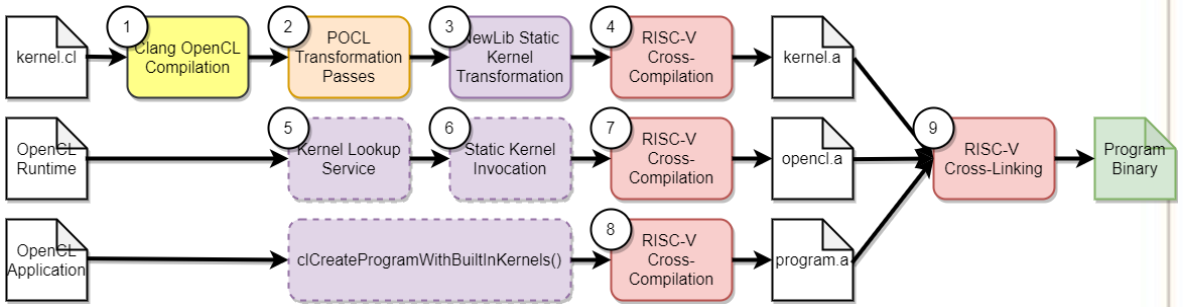
- Linux-Capable RISC-V CPUs
  - ISA extension for OS support
    - Atomics
    - Fence
    - CSRs
- Adding RISC-V support
  - Cross-compilation support for RISC-V
    - OpenCL runtime
    - OpenCL application
  - Kernel offline compilation
    - POCL binary format
    - New runtime kernel loader
    - clCreateProgramWithBinary()

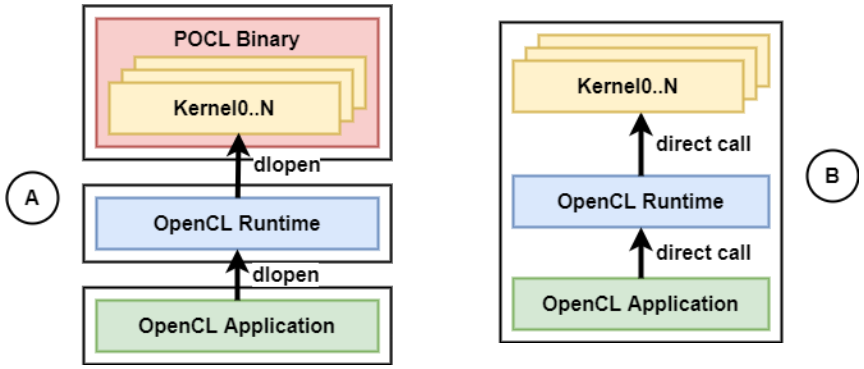




OpenCL for Newlib RISC-V CPUs

- Newlib-Capable RISC-V CPUs
  - Lowest Profile ISA Capabilities
  - Cannot host a File system nor OS
- Challenges
  - Where/how to store the kernel?
  - How to invoke the kernel at runtime?
  - What about multi-functions kernels?
- Solution
  - Single-file binary
    - OpenCL application
    - OpenCL runtime
    - OpenCL kernels
  - Use static kernel registration
    - Pre-compiled kernel static libraries
    - Runtime registration and lookup
  - clCreateProgramWithBuiltinKernels()





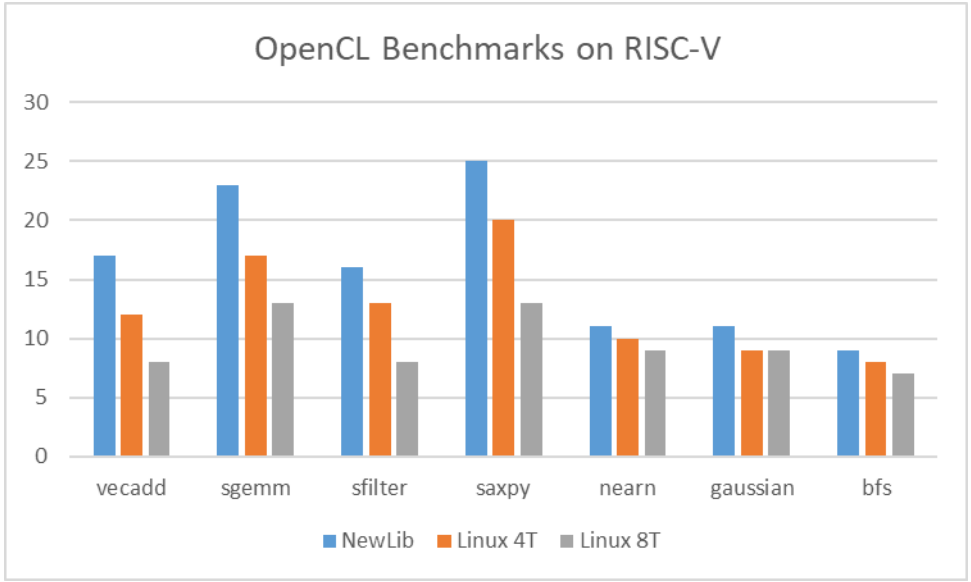
Linux (A) vs NewLib (B) Kernel Invocation Sequences

Evaluation

- Validated implementation on QEMU and Spike simulator
- Also supports the lowest RISC-V ISA profile: RV32i
- Tested Linux-capable support on
  - Fedora 64-bit OS on QEMU
  - Performance results not indicative of actual CPU scaling

RISC-V Architecture	POCL Device Class	Tested Environments
RV32im -lp32	RISCV-NewLib	Spike, QEMU
RV32imf -lp32f	RISCV-NewLib	Spike, QEMU
RV64imfd -lp64d	RISCV-NewLib	QEMU
RV32gc -lp32d	RISCV-Linux	QEMU
RV64gc -lp64d	RISCV-Linux	QEMU

Table 2: Tested RISC-V CPUs Configurations





Thank You

- Source Repository:
- https://github.com/vortexgpgpu/pocl

Thank you!