Panel Discussion:
Open Source IP – Enabling Research in Academia

OSDForum
18 September 2019
OPEN SOURCE DEVELOPER FORUM
Panel

- Moderator: Hugh Pollitt-Smith (CMC Microsystems)
- Mickaël Fiorentino (Polytechnique Montréal)
- Miodrag Bolic (University of Ottawa)
- Sebastian Magierowski (York University)
- Guy Lemieux (University of British Columbia)
- Vaughn Betz (University of Toronto)
- Lesley Shannon (Simon Fraser University)
What is CMC and what is its role?

- Not for profit – federally incorporated 1984
- Manages Canada’s National Design Network®
- Delivers micro-nano innovation capabilities across Canada

Academic and Industrial Users
LOWERING BARRIERS TO TECHNOLOGY ADOPTION

**CAD**
- State-of-the-art environments for successful design
  - Selection of high-performance Computer Aided Design (CAD) tools and design environments
  - Available via desktop or through CMC Cloud
  - User guides, application notes, training materials and courses

**FAB**
- Services for making working prototypes
  - Multi-project wafer services with affordable access to foundries worldwide
  - Fabrication and travel assistance to prototype at a university-based lab
  - Value-added packaging and assembly services
  - In-house expertise for first-time-right prototypes

**LAB**
- Device validation to system demonstration
  - Access to platform-based Microsystems design and prototyping environments
  - Access to test equipment on loan
  - Access to contract engineering services

© 2019 CMC Microsystems
Canada’s National Design Network®

A Canada-wide collaboration between 67 universities/colleges to connect 10,000 academic participants with 1,000 companies to design, make and test micro-nanosystem prototypes. CMC Microsystems manages Canada’s National Design Network®.

2018 Outcomes:
- 3655 publications
- 165 awards
- 85 patents awarded
- 555 collaborations
- 20 new startups
- 785 trained HQP moved to industry

Annually:
- 1250 connected professors
- 8850 researchers on professors’ teams
- 6200 users of computer aided design tools
- 360 physical prototypes
- 80 equipment rental items otherwise unaffordable to users

© 2019 CMC Microsystems
Industrial Supply Chain

Global partnerships to support research excellence in Canada

North America

(1) Canada
14 CAD | 8 Fab | 13 Nanofab
19 Systems & Components
42 University MNT labs

(2) USA
1 Co-operative Initiative
15 CAD | 5 Fab | 11 Nanofab
8 Systems & Components

Europe

(3) Ireland
1 Fab
(4) UK
1 CAD
1 Systems & Components
(5) France
2 Fab
1 Co-operative Initiative
(6) Sweden
1 CAD

(7) Netherlands
2 Fab
(8) Belgium
1 Fab
(9) Germany
1 CAD | 2 Fab
(10) Austria
1 Fab
Also, 1 European Co-operative Initiative

Asia

(11) China
1 Co-operative Initiative
(12) South Korea
1 Cooperative Initiative
(13) Taiwan
1 Co-operative Initiative
2 Fab
3 Systems & Components
(14) Japan
1 Co-operative Initiative
(15) Singapore
3 Fab
CMC’s role with RISC-V: Path to Implementation

- Silicon-proven design platforms
- CAD tools and flows for virtual system prototyping, FPGA, ASIC implementation, verification, PCB design
- Embedded software toolchain, software stack
- FPGA platforms for emulation & validation
- Design kits, IP for fabrication in commercial technologies
- Multi-project wafer fabrication services
- Packaging & assembly services
- Access to test equipment
- Demonstration Board
- Engineering support, consulting/design services
RISC-V Accelerator Platform Initiative

- Researchers develop

- CMC provides
  - CAD
  - Libraries
  - SDK
  - Training/support

Accelerator-Processor Interface

RISC-V Processors

FPGA/Emulator

ASIC

Demo Board
CMC Cloud FPGA/GPU Cluster

CPUs, GPUs and FPGAs in pre-validated cluster to scale heterogenous computing workloads

- Machine learning training and inference (e.g. CNN for object detection, speech recognition)
- Video Processing / Transcoding, Financial Computing, Database analytics, Networking
- Quantum chemistry, molecular dynamics, climate and weather, Genomics
- RISC-V Accelerators in Open Source Cloud Computing

### Cluster Hardware

- **Alveo U200**
- **Tesla V100**

### Cluster Configuration

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Config.</th>
<th># of nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>2 x GPU</td>
<td>3</td>
</tr>
<tr>
<td>4-6</td>
<td>2 x FPGA</td>
<td>3</td>
</tr>
<tr>
<td>7,8</td>
<td>1 x GPU</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1 x FPGA</td>
<td></td>
</tr>
</tbody>
</table>

### 1 Node Specifications

- Dual 12 core 2.2-to-3.0 GHz CPU
- 192 GB RAM
- 300 GB local storage
- 100 Gb EDR node interconnect
- 10 GbE storage network

### FPGA/GPU cluster Specifications

© 2019 CMC Microsystems
Software stack for the FPGA/GPU cluster

Applications
- Machine Learning
  - Caffe
  - TensorFlow
  - MXNet
- DB
- Video
- Finance
- Genomic

ML Framework
- cuDNN
- MLOpen
- BLAS, FFT, RNG
- C/C++
- Python
- OpenCL

Middleware, Tools and Libraries
- Runtime

Hardware
- Hardware Accelerators

Hardware Accelerators
- Multicore Host CPU
- PCIe

© 2019 CMC Microsystems
Panel

- Mickaël Fiorentino (Polytechnique Montréal)
- Miodrag Bolic (University of Ottawa)
- Sebastian Magierowski (York University)
- Guy Lemieux (University of British Columbia)
- Vaughn Betz (University of Toronto)
- Lesley Shannon (Simon Fraser University)
Panel Presentations

With respect to Open Source HW and/or SW, describe:

1. What you’ve done in the past
2. What you’re doing now, and
3. What you need from the ecosystem/community
How We Use RISC-V: *Research*

**KeyV**: A Self-Timed *In-Order* RISC-V Processor

[https://git.cmc.ca/mickael.fiorentino/keyv.git](https://git.cmc.ca/mickael.fiorentino/keyv.git)

**Principles**
- Microarchitectural concepts that can *lower the power consumption*
- Emphasis on clocking mechanisms & instruction level parallelism

**Cores** — *RV32IM*
- *KeyV*: Self-Timed Core
- *SynV*: Baseline Synchronous Core

**Implementation**
- 45nm GPDK — *Cadence*
- 65nm LP — *TSMC*
**How We Use RISC-V: Teaching**

**mini-riscv**: From Specification To Layout

https://git.cmc.ca/mickael.fiorentino/ele8304.git

**Design flow**
- RTL design – VHDL-2008
- 45nm GPDK – Cadence
- Simulations – Modelsim
  Behavioral, post-synthesis, post-pnr
- Synthesis – Genus
  Timing analysis, Power analysis
- Place & Route – Innovus
  DRC, LVS, STA

**Specifications**
- Subset of RV32I: 25 instructions
- Basic memory model

**Software**
- ASM Fibonacci benchmark
  riscv32-unknown-elf-gcc -O0

**Microarchitecture**
- 5-stages pipeline
- Basic branch prediction
Leverage The (CMC) **Ecosystem**

**Toolchain**
- `source /CMC/scripts/riscv.rv32ima.csh`

**Verification**
- Software models
- Verification IPs (UVM, UVVM)
- Formal verification

**Design reuse**
- Shared library of standard building blocks (FIFOs, BUS, Cores, etc.)

**Benchmarks**
- Standard *Cores* that we can compare against
- Figure of merit: Coremark post-synthesis, 45nm GPDK?

**Open Source (Hardware) Design Environment**
- Open Source PDK – 45nm
- Open Source CAD – Simulation, (HLS) Synthesis, Place & Route...

**Training**
- Online tutorials
- Webinars
- Events & Meetups

**Collaboration**
- `git.cmc.ca`
- `nextcloud.cmc.ca`
- `matrix.cmc.ca`
Open source tools in education

Miodrag Bolic
Professor
School of Electrical Engineering and Computer Science
University of Ottawa
mbolic@uottawa.ca
Open source in education

• Reasons for using open source in education
  – Improving the teaching method
    • provide students experimentation resources that help them to understand abstract concepts of science and engineering.
  – Reducing costs
  – Promoting students’ engagement
  – Fostering students’ creativity
    • students can move from simple technology users to the more active role of makers.

• Open source in education
  – Software
  – Hardware
  – Simulators
  – Operating systems
  – Instruction Set Architectures (it is estimated that more than 500 universities will teach RISC-V courses by next year)
RISC-V in Computer System Design Course

- 3rd year undergraduate in computer engineering in 2018
- Tools: Bluespec
- Labs
  - RISC-V Multi-Cycle and Two-Stage Pipeline
  - RISC-V 6-stage Pipeline and Branch Prediction
  - RISC-V with Caches
- Material: MIT course Complex Digital Systems by Prof Arvind
- Experience
  - Assistance from MIT was amazing
  - However, there was no appropriate support for students
- Need
  - Training for graduate students in developing applications for RISC-V as well as in developing new architectures and testing them in FPGAs
  - These students will then be able to transfer their knowledge to undergraduate students
Open-source resources in my other courses

- **Simulators**
  - Parallel processing architectures course – 4th year undergraduate
  - Multi2sim
    - Support for CPU and GPU architectures
    - Students simulated multi-core CPU and GPU and ran code on the simulated platforms
    - No support

- **Open Source Hardware Project**
  - Parallella multi-processing board
    - Students developed software accelerators but did not really utilize the open hardware
    - Limited software, problems with debugging, ...

- **Open source software**
  - Machine learning graduate course
    - Python and Jupyter notebook for teaching
    - Open-source software approach – everything is on Github
Done in Past

- FPGA Acceleration
  - embedded bioinformatics
    - DNA sequencing

- Open-source
  - RIFFA
    - Reusable Integration Framework FPGA Accelerators
  - HW (code) for PCIe
  - SW (API) for comms
    - fpga_send(): CPU → FPGA
    - fpga_recv(): CPU ← FPGA
Doing Now

- The same…
  - embedded bioinformatics
- …but with ASICs

- Open-source
  - want this beside RISC-V
  - rapid prototyping with open source hardware description
    - Chisel
    - PyMTL
What’s Needed From Community?

• I/O is a big deal
  – targeting real-time apps
  – FPGA currently
    • 0.8 GB/s (full duplex)
    • easy for PCIe 2.0
  – FPGA planned
    • 3.5 GB/s
    • not as easy for PCIe 2.0

• RIFFA to 3.0 & Ultrascale+
  – PCIe 4 & 5?

• FPGA SoCs
  – ARM w/i OS & AXI stream
    • tight accelerator integration

• ASICs
  – how to integrate ASICs with RISC-V?
    • many IP blocks
  – I/O for accelerated RISC-V?
    • board design for + GB/s?
    • northbridge
  – Linux on RISC-V ASIC?
    • getting complex systems up and running
  – HLS for ASICs?
    • Catapult, Stratus
Guy Lemieux
Prof. @ UBC
CEO @ VectorBlox
Past IP ...

• UofT (myself)
  • Scalable high-frequency counters

• VectorBlox (not open source)
  • MXP Matrix Processor
    • CNN Accelerator for MXP
    • DNN Accelerator for MXP
  • ORCA RISC-V Processor (open source)
    • Binary NN Accelerator for ORCA (not open source)

• UBC (Dr. Ameer Abdelhadi)
  • Multi-port RAM
  • Binary CAM
  • Ternary CAM
Present IP...

• More RISC-V
  • New CPU design optimized for FPGAs
  • Tools for building systems
  • Peripherals, accelerators

• Manycore array

• OpenVX acceleration system for FPGAs
Help from ecosystem/community

• RISC-V IP
  • Debug/JTAG controller (lightweight) + software
  • Peripheral library (PCIe endpoint + DMA scatter/gather, USB, Ethernet, UART, I2C, bus interfaces + bridges, etc)

• System tools
  • Build processor + peripherals + memory
  • Multi-vendor support (all FPGAs)
  • Simulation
  • Software “board dev kits” (header files, linker scripts, etc)

• Custom PCB fabrication as easy as FPGA design

• Contributions
  • Companies can donate $$$ or time
  • Individuals can donate time (esp. support/maintenance) because it’s hard for researchers
A Cautionary Tale...

- Selling Hardware IP is not a great business model
  - Skeptical customers
  - Escalating costs for verification/support/marketing
  - Difficult to get credible testimonials
  - Difficultly getting traction with large partners
  - Nearly every single “IP company” actually makes more money on services

- Maybe Hardware IP should be free/open?
  - Selling it isn’t a valid business model anyways

- Large vendors must provide better ecosystem + support for Hardware IP (open or not)
  - Think of the “App Store”, not “Frequent Flyer Miles”
  - Large vendors will always make money on hard product sales
Open Source CAD Tools & FPGA Hardware

Vaughn Betz
Professor, Dept. of ECE
NSERC/Intel Industrial Research Chair in Programmable Silicon
Faculty Affiliate, Vector Institute

vaughn@eecg.utoronto.ca
Past: Open Source FPGA CAD for Research

- VPR / VTR: Open source CAD flow for FPGAs
  - Research new architecture ideas
  - Research new CAD algorithms
  - Successfully commercialized: became P & R engine of Quartus and Intel’s FPGA arch.
    Exploration tool
Present: Open Program and Generate Hardware

• **Research** still big focus
• Completing the stack
  – Multiple FPGA start-ups & Darpa POSH projects using as production CAD tools
  – Symbiflow/Google: programming commercial Lattice & Xilinx devices
  – Custom FPGA layout & bitstream generators

Image source: Grady & Anderson
Future: Open Research & Production Flow?

• Ideal: open compiler for FPGAs (gcc-like)
• Barriers
  – Closed commercial bitstreams
    • Need to reverse engineer → slow, hard
    • Smaller vendors → open bitstream to gain mindshare?
  – Architects / reviewers to keep tool healthy
    • Complex tools → need architects & code reviewers
    • Always busy / graduating / in short supply
    • Can industry or research consortium fund?
    • Community building & mentoring?
  – Forking
    • Grad student: fastest way to a paper is fork code & change
    • Hobbyist: more fun & immediate results via recode everything
    • But redundant work, slower progress!
    • How to incentivize / organize work going back to trunk?
TAIGA RISC-V 32-bit CPU

By
Lesley Shannon
School of Engineering Science
Simon Fraser University

OSD Forum Panel
Past Work

- Polyblaze multicore softprocessor
  - Not open source (limited by others)

- Performance monitoring
  - Limited by not having platforms to demonstrate work on

- APIs/middleware for abstracting, scheduling and resource management of heterogeneous systems
  - Limited by not having platforms to demonstrate work on
Current Work
What we need

- Help with Verification
  - *Benchmarks, Compliance tests*, tools, frameworks, methods (formal?)

- Compiler/Synthesis tool support and Design Environments
  - Easy wrappers to integrate and use

- Credit
  - Research requires $$

**Key Point:** Researchers cannot guarantee correctness and we do not have access to the frameworks that would allow us to do so.