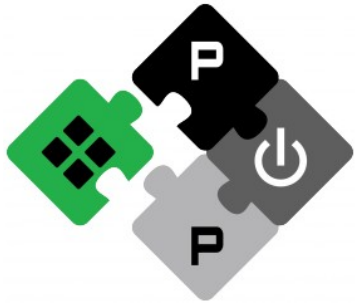


PULP PLATFORM
Open Source Hardware, the way it should be!

Banshee: A Fast LLVM-Based RISC-V Binary Translator

ICCAD 2021

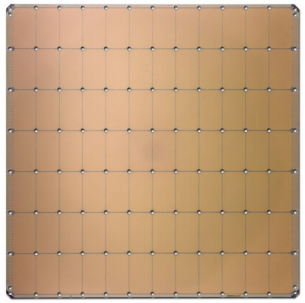
Samuel Riedel (sriedel@iis.ee.ethz.ch)
Fabian Schuiki (fschuiki@iis.ee.ethz.ch)
Paul Scheffler (paulsc@iis.ee.ethz.ch)
Florian Zaruba (zarubaf@iis.ee.ethz.ch)
Luca Benini (lbenini@iis.ee.ethz.ch)

ETH zürich

 http://pulp-platform.org  @pulp_platform  https://www.youtube.com/pulp_platform

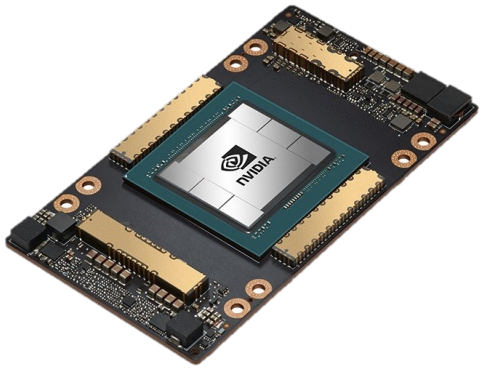
Multi-core and manycore systems

- **Crucial for modern workloads**
  - Machine learning
  - Computational photography
- **RISC-V[2] popular in research and industry**
  - Open and modular ISA
- **Open-source systems**
  - Celerity[3], Manticore[4], MemPool[5]



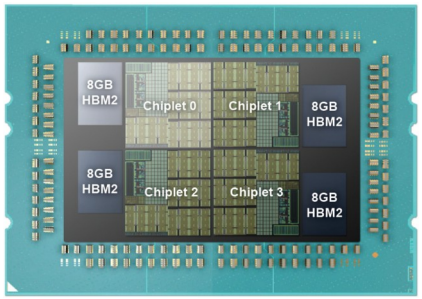
Cerebras WSE-2
850'000 Cores

https://cerebras.net/chip/



NVIDIA A100: 6912 Cores

https://www.nvidia.com/de-de/data-center/a100/

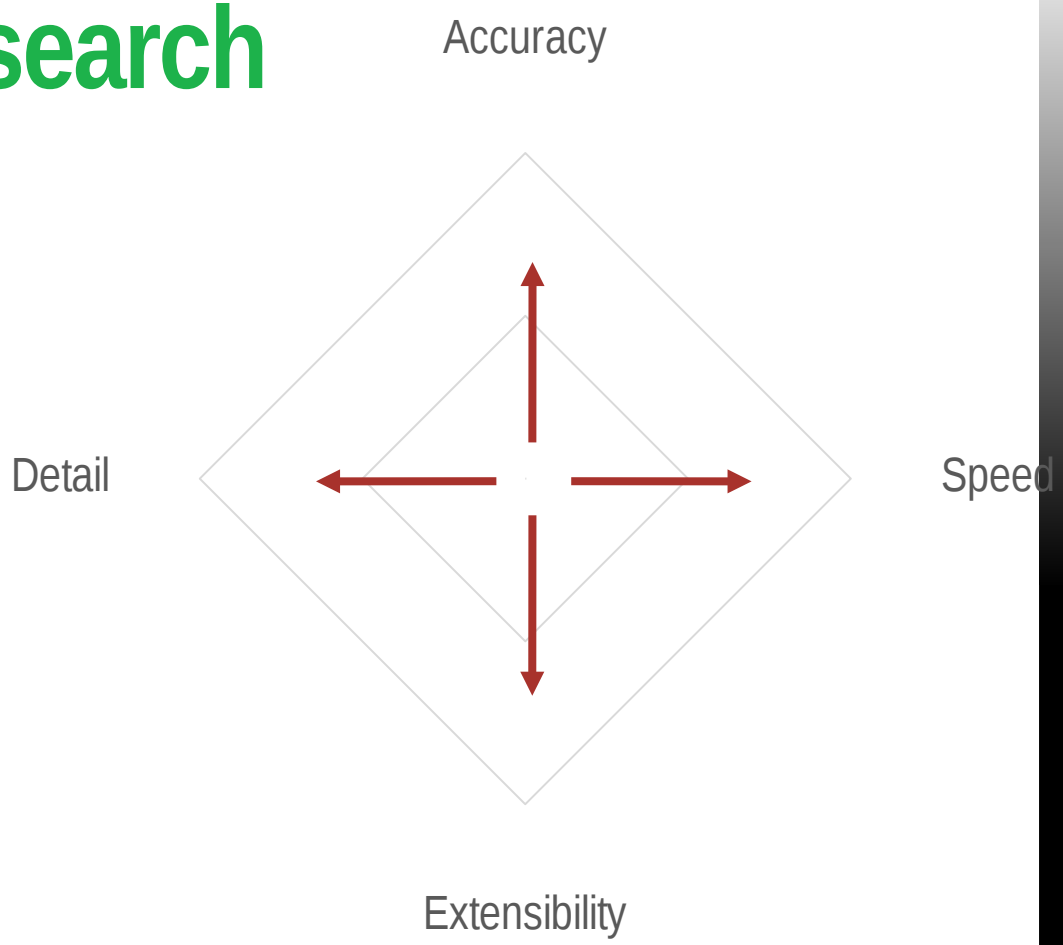


Manticore
4096 Cores

https://arxiv.org/pdf/2008.06502.pdf

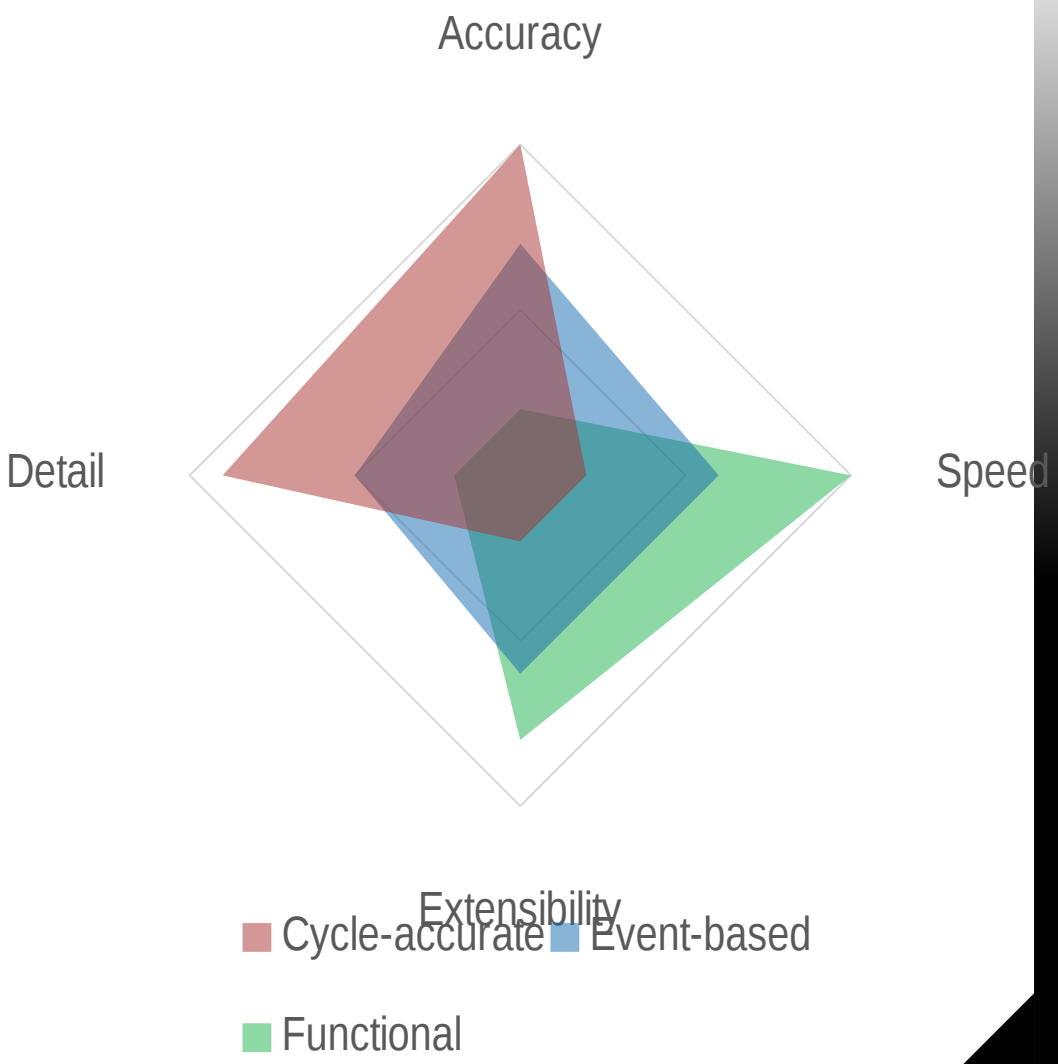
Computer architecture research

- **Simulation and emulation tools**
  - Exploration
  - Performance estimation
  - Verification
  - Software development
- **Conflicting design goals**[1]
  - There is not '*one to rule them all*'



Types of simulators

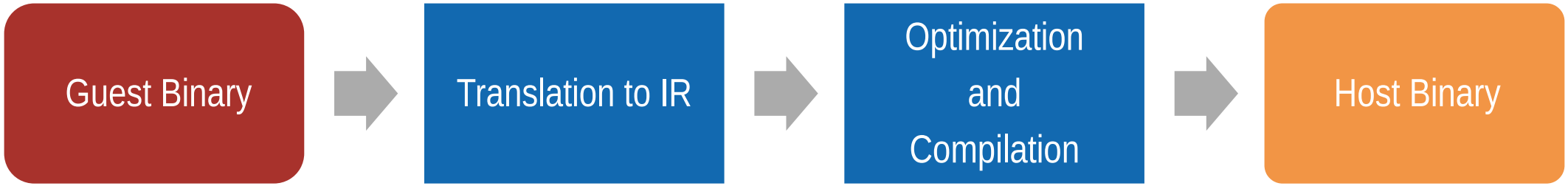
- **Cycle-accurate**
  - Verilator[7], Questa Advanced Simulator[8]
  - Very slow for large systems (tens of kIPS)
- **Event-based**
  - gem5[9], GVSoC[10]
  - Few MIPS, still too slow
- **Functional**
  - QEMU[11,12], rv8[13], R2VM[24]
  - Few GIPS, but only for a handful of cores
- **None are well-suited for manycore!**



Banshee

- **Designed to simulate hundreds of 32-bit RISC-V cores**
  - Generic, cluster-based architecture
  - Easy to extend (peripherals, ISA extensions)
  - Explore architectures quickly
- **Functional simulator**
  - Develop and debug software
  - Fast simulation
  - Instruction accurate
  - Lightweight extension for performance estimation
- **Open-source:** https://github.com/pulp-platform/snitch/tree/master/sw/banshee

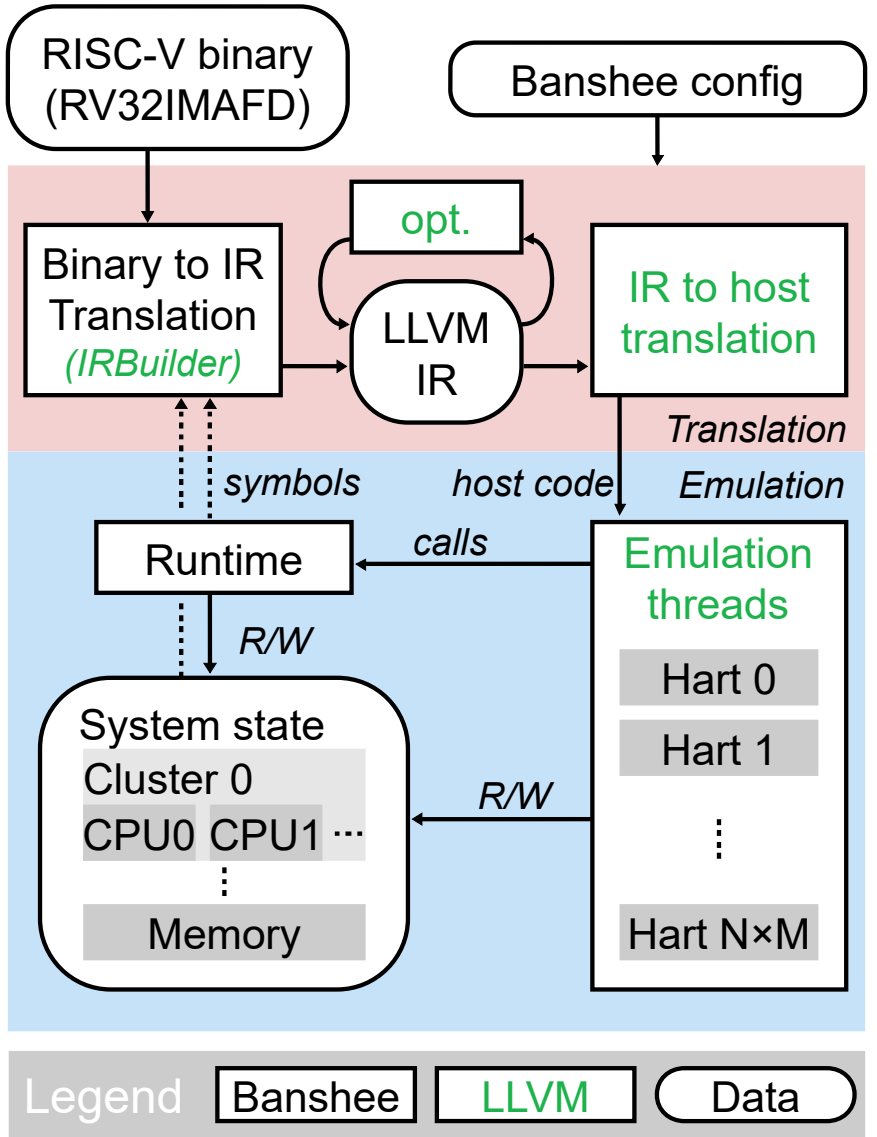
Banshee uses static binary translation



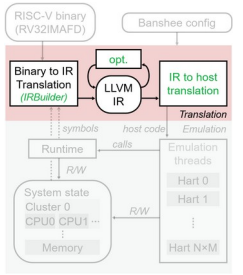
- **Static**
  - Compile entire binary before execution
  - No self-modifying or relocatable code
  - More optimization space
- **Dynamic**
  - Only translate necessary code
  - Supports self-modifying and relocatable code
- **Banshee aims to simulate compute-centric manycore accelerators → Static binary translation**

Banshee overview

- **Inputs**
  - RISC-V guest binary
  - Configuration file
- **Translation creates host binary**
  - Using the LLVM infrastructure
- **Emulation runs the host binary simulating the target architecture**
- **The runtime links the translation and emulation**



Translation to IR

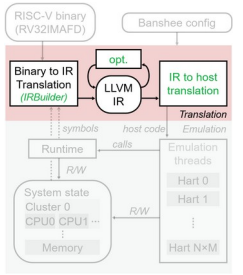


- **Map registers to memory accesses**
  - LLVM will promote them back to host registers
- **For each instruction:**
  - Emit stores to get the register values
  - Translate instruction to LLVM IR
  - Write back to the register
- **Compile and optimize with LLVM**

```
// RISC-V ASM
add a0, a1, a2
```

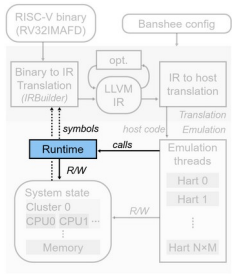
```
; LLVM IR
%v1 = load i32, i32* %ptr_a1
%v2 = load i32, i32* %ptr_a2
%v0 = add i32 %v1, %v2
store i32 %v0, i32* %ptr_a0
```

Translating memory access



- Load and store instructions
- Utilize host's memory hierarchy
- Fast path
  - Map frequently used memory directly to the host's heap
  - `%ret = load i32, i32* %addr`
- Slow path
  - Model memory-mapped resources (peripherals) as runtime calls
  - `%ret = call i32 @runtime_load(i32* %addr)`

Runtime

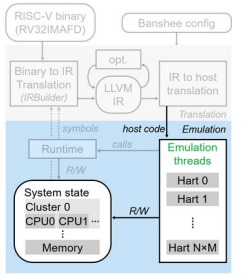


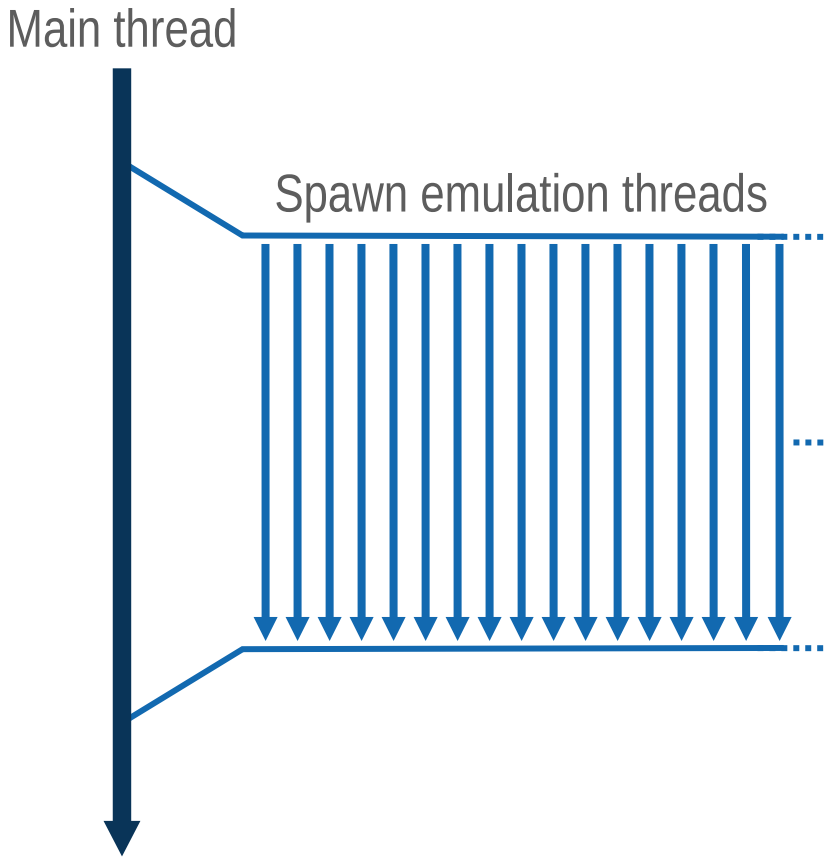
- **Use runtime calls for complex instructions**
  - Emitting complex IR is tedious
  - Implement the behavior in a high-level language
- **Runtime also provides access to global state**

```
int runtime_load(addr)
  if addr in control register
    // Implement control register
  if addr in peripheral
    // Handle peripheral read
  if addr in global memory
    // Read from shared memory
```

Emulation

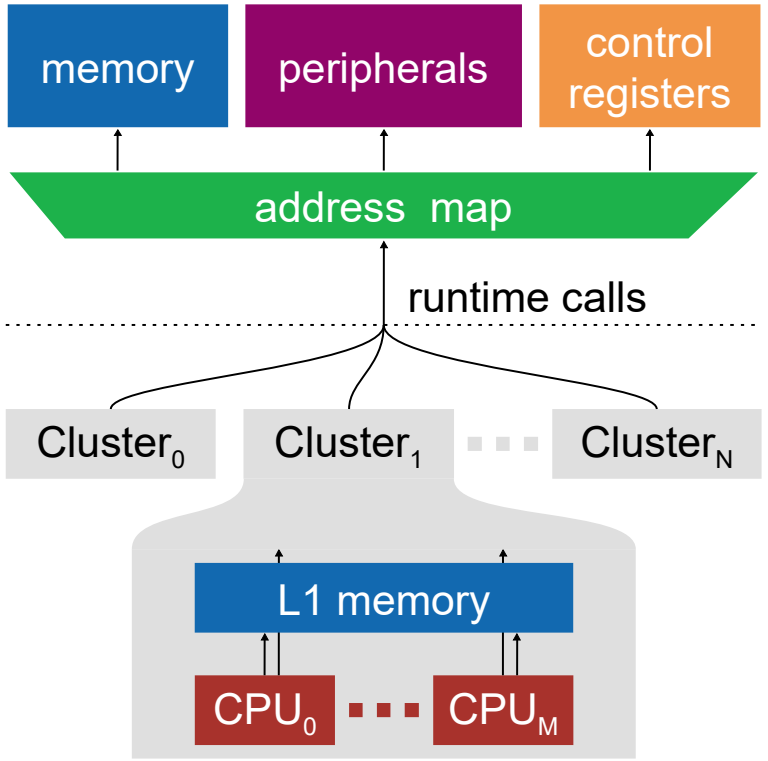
- **Use host's parallelism**
  - Spawn one thread per guest core
- **Reduce synchronization**
  - Minimize blocking runtime calls
  - Do not model shared effects like bus contention or shared caches





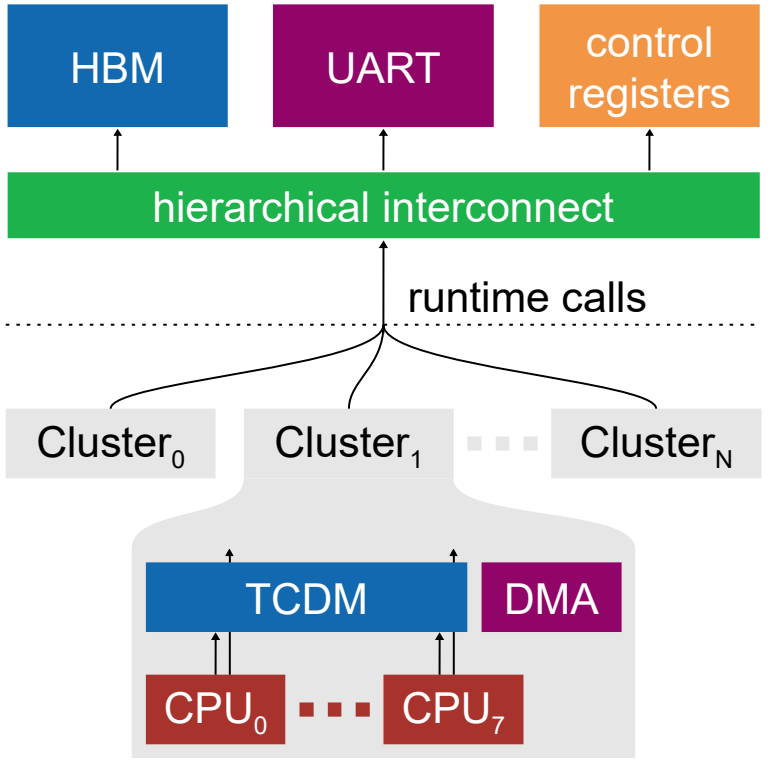
Generic Architecture

- **Clustered architecture**
  - Private L1 memory allocated in heap
  - Configurable number of cores
- **Memory hierarchy modeled as runtime calls**
  - Easy implementation of peripherals



Manticore[4]

- Features 4096 RV32IMAFD cores
- Clustered design
  - 8 cores per cluster
  - Shared tightly-coupled data memory
  - One DMA per cluster modeled
- Supports various ISA extensions

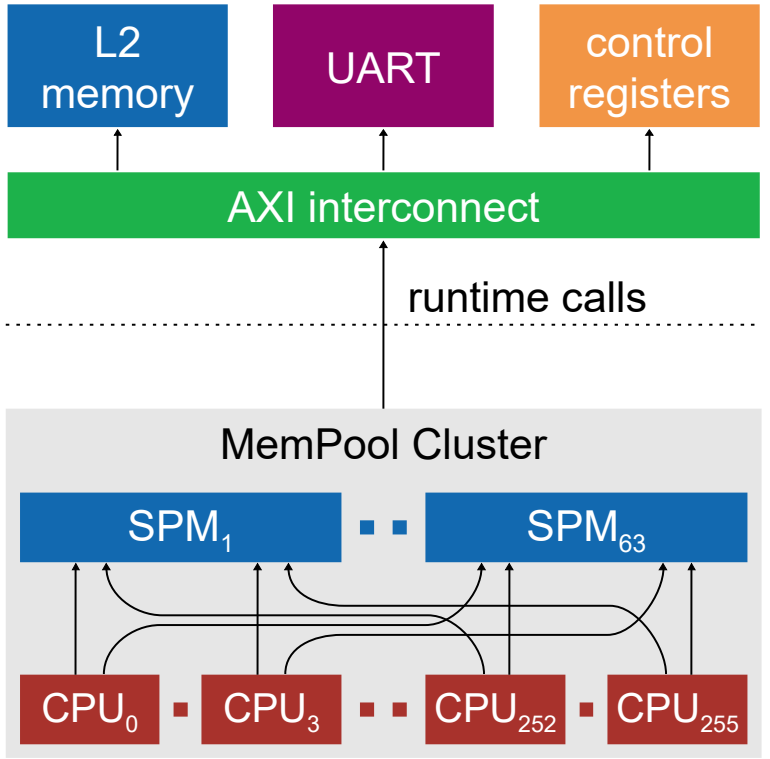


ISA Extensions

- **Floating-point repetition (Xfrep[30])**
  - Repeats a sequence of FP instructions to eliminate branches
  - Handled during translation
- **(Indirect) Stream Semantic Registers (Xssr[16] & Xissr[31])**
  - Map a programmable memory stream to registers
  - Eliminates explicit load and store instructions
  - Address generation is implemented in the runtime

MemPool [5]

- 256 RV32IMA cores
- Single cluster design
- NUMA scratch-pad memory
  - 1, 3, or 5 cycles latency depending on distance
- AXI interconnect to L2 memory and peripherals



Performance estimation

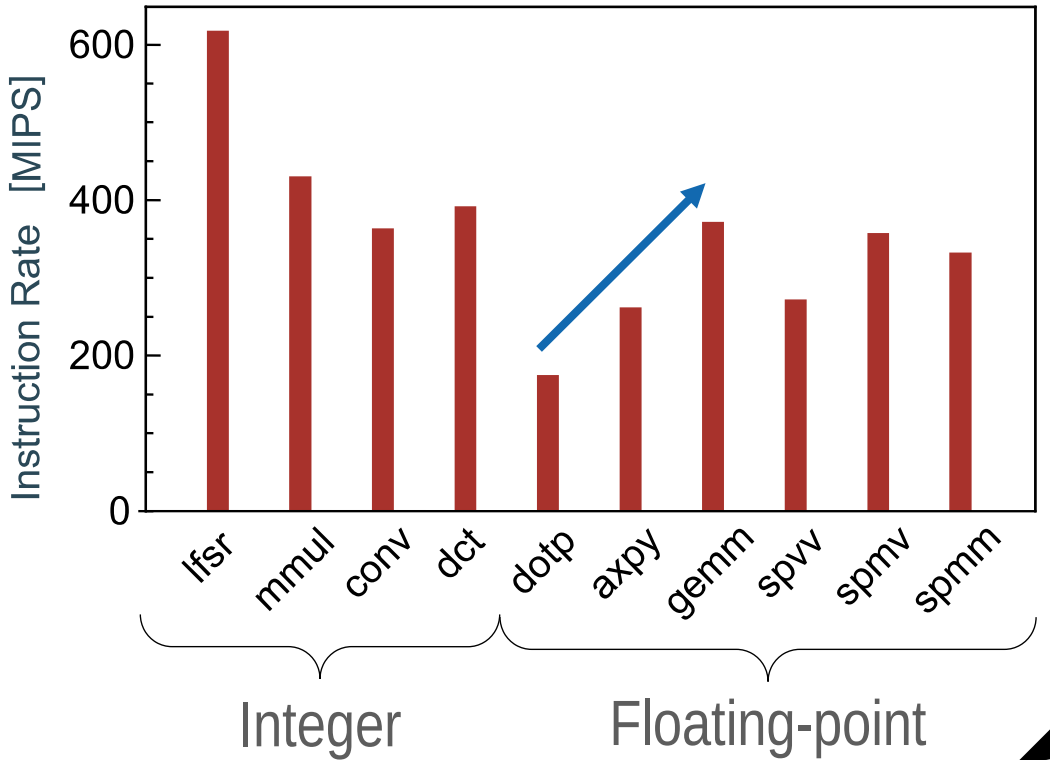
- Instruction count as first-order approximation
- Optionally model read-after-write stalls
  - Instruction latencies
  - Load latency (per memory region)
  - Scoreboard keeps track latencies
- Relies solely on private data
  - No shared effects are modeled like bus contention or shared caches

Evaluation Setup

- **Host System**
  - Two 64-core AMD EPYC 7742 CPUs @ 2.25 GHz
  - No simultaneous multithreading
  - 256 MiB of L3 cache
  - 512 GiB system memory
- **Each kernel runs 10k iterations to minimize static overhead**
- **All reports are averages of 10 runs**

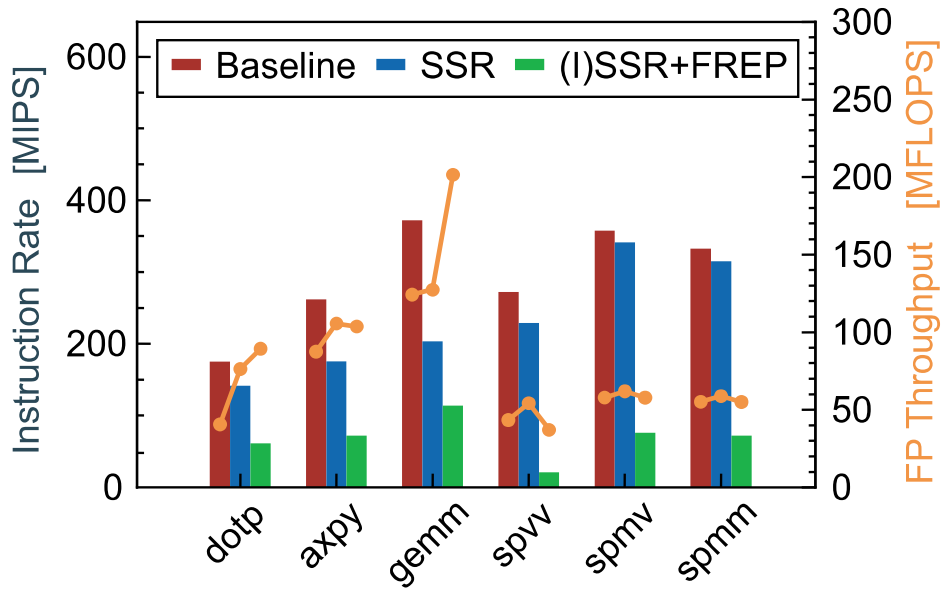
Single Core Performance

- **LFSR achieves 618 MIPS**
  - Fully compute bound
- **Average performance:**
  - Integer kernels: 396 MIPS
  - Dense FP kernels: 372 MIPS
  - Sparse FP kernels: 358 MIPS



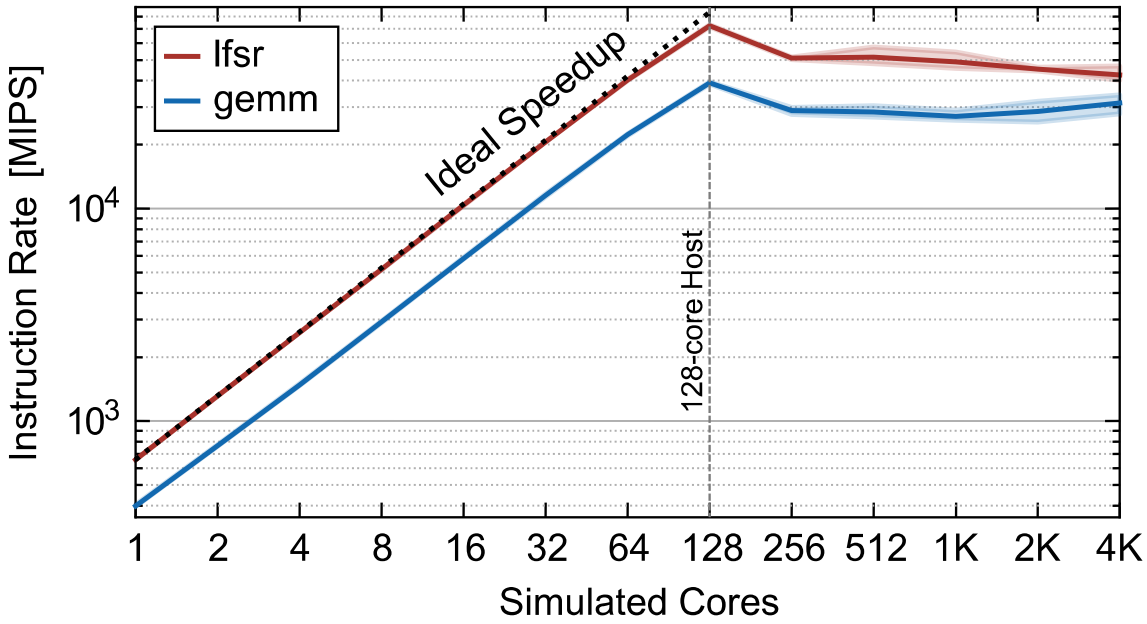
ISA Extension Performance

- Instruction rate drops with ISA extensions
  - Due to increase in simulated FP instructions
- Simulated FP instructions increase for dense kernels
- Sparse kernels do not show the same benefit
  - Costly runtime calls
  - Random memory accesses



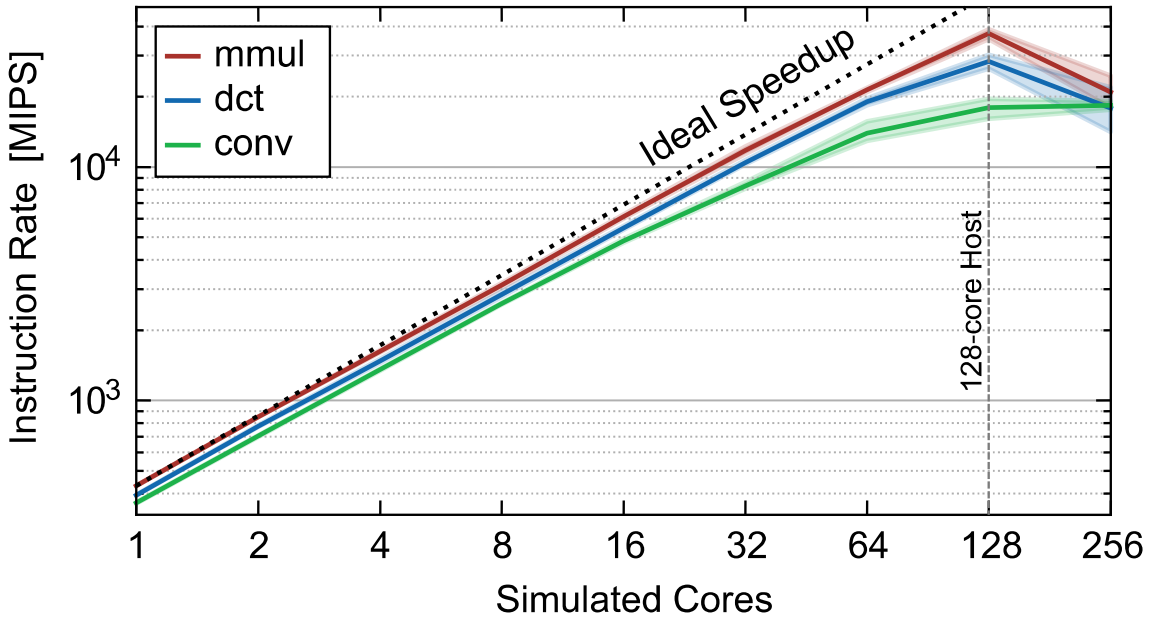
Scaling with Manticore

- Close to ideal speedup
- Very good scaling
  - LFRS: 72 GIPS
  - GEMM: 40 GIPS
- Instruction rate flattens after guest cores overtake host cores
  - But stays constant



Scaling with MemPool

- Single cluster also scales
  - Peak performance: 37 GIPS
- Shared L1 memory leads to more cache conflicts in host



Comparison to related work

Simulator	Method	Single-core [MIPS]	Multi-core [MIPS]	Cores
gem5[19]	Event-based	0.2	1.2	8
Sniper*[20]	Interval-based	0.5	2.0	16
GVSoC[10]	Event-based	2.5	20.0	8
QEMU[24]	DBT	269.0	1’076.0	4
R2VM[24]	DBT	413.0	1’652.0	4
riscvOVPsim[25]	DBT	1’000.0	-	-
Banshee (ours)	**SBT**	**618.0**	**72’397.0**	**128**

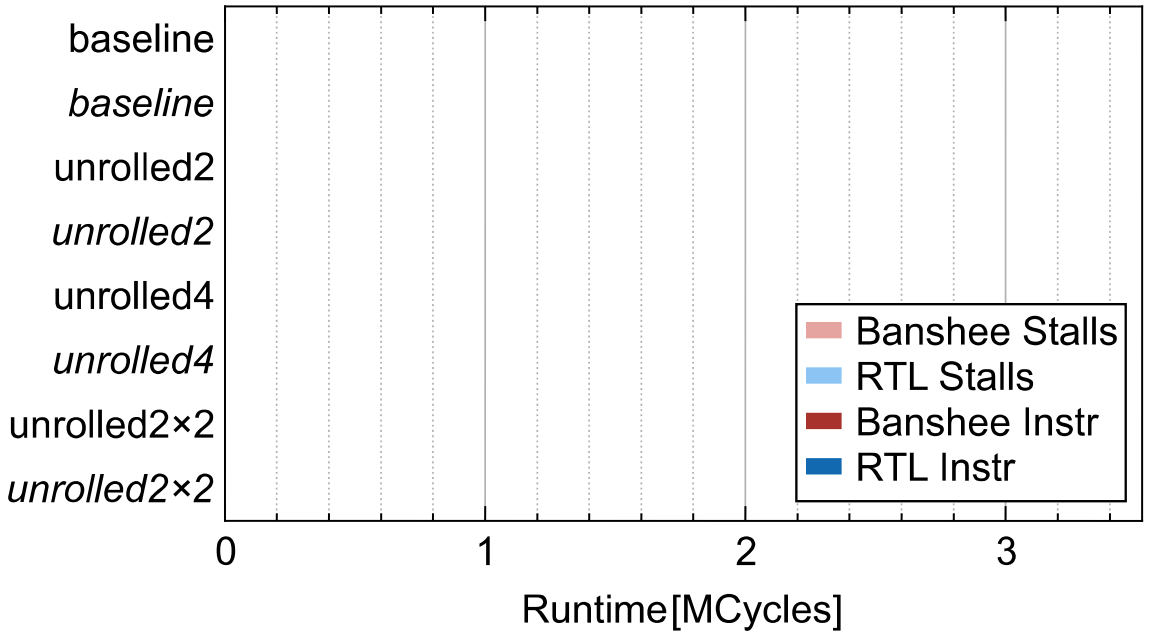
1.5x

44x

*All results are from RISC-V emulation except Sniper is from x86.

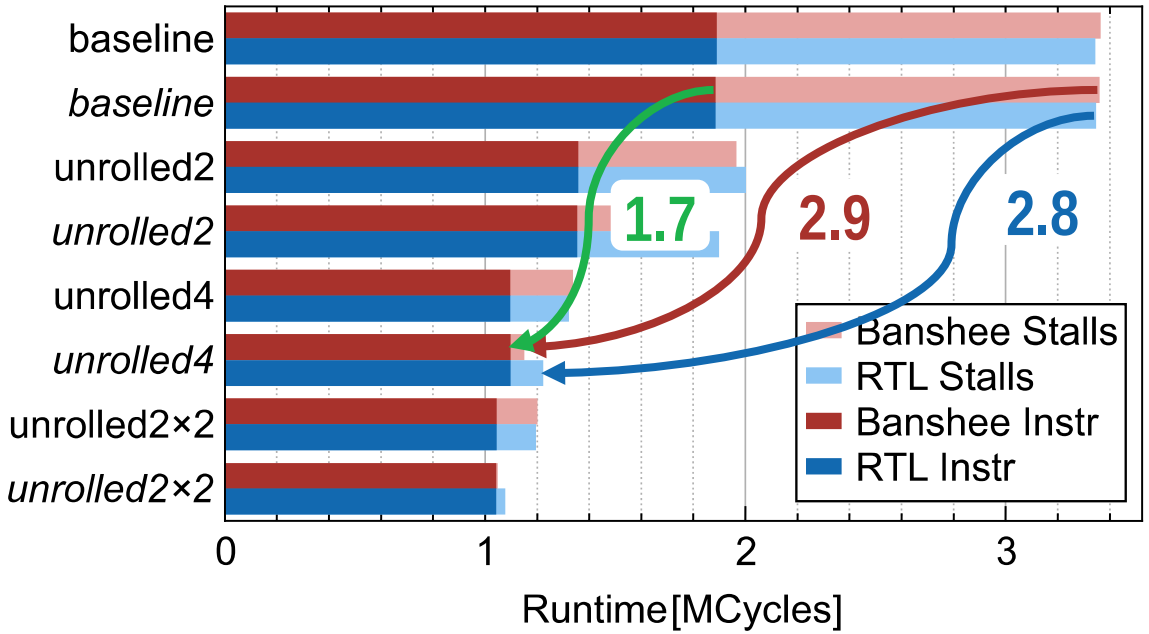
Latency modeling evaluation

- **Evaluate *mmul* implementations on MemPool**
  - Heavily relies on hiding latencies
  - Use different ways of loop unrolling
- **Instruction scheduling in compiler (*in italic*)**
- **Compare to stalls in RTL simulation**



Latency modeling results

- Instruction count matches
- Better runtime estimation than instruction count
- Runtime estimation within 2% on average
- Adds 12% runtime overhead



Conclusion

- **Banshee is:**
  - Fast: Up to 618 MIPS (single core) and 72 GIPS (128 cores)
  - Extensible: Implemented various custom ISA extensions and peripherals
  - Instruction accurate
- **Evaluated on open-source systems**
  - Manticore (4096 cores) and MemPool (256 cores)
- **Facilitates architecture exploration**
- **Helps developing software**
- **Open-source:** https://github.com/pulp-platform/snitch/tree/master/sw/banshee

References

[1] A. Akram and L. Sawalha, "A Survey of Computer Architecture Simulation Techniques and Tools," IEEE Access, vol. 7, pp. 78120–78145, 2019.

[2] A. Waterman and K. Asanovi´c, "The RISC-V Instruction Set Manual, Volume I: User-Level ISA," tech. rep., RISC-V Foundation, 2019.

[3] S. Davidson, S. Xie, C. Torng, K. Al-Hawai, A. Rovinski, T. Ajayi, L. Vega, C. Zhao, R. Zhao, S. Dai, A. Amarnath, B. Veluri, P. Gao, A. Rao, G. Liu, R. K. Gupta, Z. Zhang, R. Dreslinski, C. Batten, and M. B. Taylor, "The celerity open-source 511-core risc-v tiered accelerator fabric: Fast architectures and design methodologies for fast chips," IEEE Micro, vol. 38, no. 2, pp. 30–41, 2018.

[4] F. Zaruba, F. Schuiki, and L. Benini, "Manticore: A 4096-core RISC-V Chiplet Architecture for Ultra-efficient Floating-point Computing," IEEE Micro, vol. 41, no. 2, pp. 36–42, 2020.

[5] M. Cavalcante, S. Riedel, A. Pullini, and L. Benini, "MemPool: A Shared-L1 Memory Many-Core Cluster with a Low-Latency Interconnect," in 2021 Design, Automation, and Test in Europe Conference and Exhibition (DATE), (Grenoble, FR), pp. 701–706, mar 2021.

[7] W. Snyder, "Verilator," 2021. Available at https://www.veripool.org/verilator.

[8] Siemens, "Questa Advanced Simulator — Siemens Digital Industries Software," 2021. Available at https://eda.sw.siemens.com/en-US/ic/questa/simulation/advanced-simulator.

[9] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A.Wood, "The gem5 simulator," ACM SIGARCH Computer Architecture News, vol. 39, pp. 1–7, may 2011.

[10] E. F. Zulian, G. Haugou, C. Weis, M. Jung, and N. Wehn, "System simulation with pulp virtual platform and systemc," in Proceedings of the Conference on Rapid Simulation and Performance Evaluation: Methods and Tools, RAPIDO '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[11] F. Bellard, "QEMU, a fast and portable dynamic translator," in Proceedings of the annual conference on USENIX Annual Technical Conference, 2005.

[12] E. G. Cota, P. Bonzini, A. Bennee, and L. P. Carloni, "Cross-ISA machine emulation for multicores," in 2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), pp. 210–220, feb 2017.

[13] M. Clark and B. Hoult, "rv8: a high performance RISC-V to x86 binary translator," in Workshop on Computer Architecture Research with RISCV (CARRV), vol. 7, 2017.

[16] F. Schuiki, F. Zaruba, T. Hoefler, and L. Benini, "Stream Semantic Registers: A Lightweight RISC-V ISA Extension Achieving Full Compute Utilization in Single-Issue Cores," IEEE Transactions on Computers, vol. 70, pp. 212–227, feb 2021.

[19] T. Ta, L. Cheng, C. Batten, and T.-p. Runtimes, "Simulating Multi-Core RISC-V Systems in gem5 Task-Parallel System Design Space Exploration," in Workshop on Computer Architecture Research with RISC-V (CARRV), 2018.

[20] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation," in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '11, (New York, New York, USA), ACM Press, 2011.

[24] X. Guo and R. Mullins, "Accelerate Cycle-Level Full-System Simulation of Multi-Core RISC-V Systems with Binary Translation," in Workshop on Computer Architecture Research with RISC-V (CARRV), may 2020.

[25] Imperas Software Ltd., "riscvOVPsim - Free Imperas RISC-V Instruction Set Simulator," 2021. Available at https://www.imperas.com/riscvovpsim-free-imperas-risc-v-instruction-set-simulator.

[30] F. Zaruba, F. Schuiki, T. Hoefler, and L. Benini, "Snitch: A tiny Pseudo Dual-Issue Processor for Area and Energy Efficient Execution of Floating-Point Intensive Workloads," IEEE Transactions on Computers, feb 2020.

[31] P. Scheffler, F. Zaruba, F. Schuiki, T. Hoefler, and L. Benini, "Indirection Stream Semantic Register Architecture for Efficient Sparse-Dense Linear Algebra," in 2021 Design, Automation, and Test in Europe Conference and Exhibition (DATE), 2021.