Building a Heterogeneous ISA Research Platform Using the OpenPiton Framework

Katie Lim
Princeton University

David Wentzlaff
Princeton University

With the approaching end of Moore's Law and the end of Dennard scaling, computer architects must look for ways to more effectively use available transistors while respecting power and energy constraints. Researchers have begun to explore heterogeneous systems as a possible solution to this problem. These systems seek to improve their energy efficiency by catering to the fact that applications may have different performance requirements.

Chips with heterogeneous microarchitecture, such as ARM's big.LITTLE architecture [2] and Apple's A11 processor [1], have become common in mobile processors for this reason. By using the smaller, lower-power cores for applications that do not need the larger cores' computational power, these processors achieve better energy efficiency without impacting the user's experience. More recent work by Venkat et al. [7] explored microprocessors with heterogeneous ISAs and heterogeneous microarchitecture in simulation. Their work found that these systems can offer additional performance and energy benefits.

Heterogeneous ISA systems pose interesting challenges to system software due to differing per-ISA program state. DeVuyst et al. [5] developed a compiler which generates binaries that can be migrated between ISAs. The researchers behind Popcorn Linux have looked at migrating binaries during runtime [6] and what an OS might look like running on a multi-ISA system [4]. They evaluated an x86-ARM system where the cores were connected over PCI, but the system did not have shared memory, which meant that migration of binaries during execution was extremely expensive. However, it is difficult to create a shared-memory heterogeneous ISA multiprocessor from off the shelf parts without significant engineering efforts.

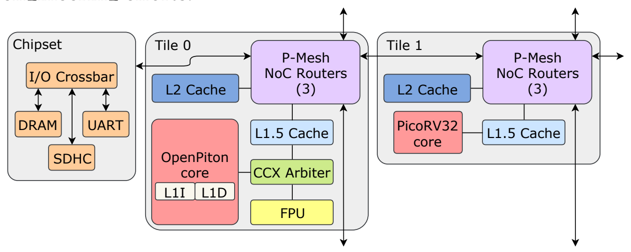

Figure 1: Diagram of a two-core design with an OpenPiton tile and a PicoRV32 tile (Derived from [3])

To assist heterogeneous ISA research, we built an open-source FPGA implementation of a heterogeneous ISA, heterogeneous microarchitecture, shared memory multiprocessor. We integrated a small multicycle RISC-V core called PicoRV32 [8] with OpenPiton [3]. OpenPiton is an open-source manycore framework. It is a tile-based manycore using a SPARC core with a 6-stage in-order pipeline. A comparison of the cores is presented in Table 1.

	OpenPiton	PicoRV32
ISA	SPARC v9	RISC-V I
Word size	64 bits	32 bits
Endianness	Big endian	Little endian
Implementation	6-stage in-order pipeline	Multicycle
L1 Cache	Yes	No
MMU	Yes	No
FPU	Yes	No

Table 1: Comparison of the OpenPiton core and the PicoRV32 core

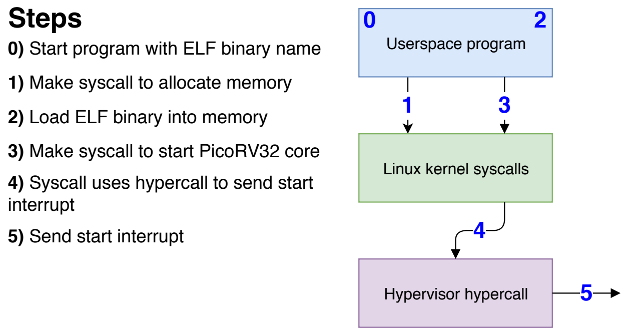
Steps
0) Start program with ELF binary name
1) Make syscall to allocate memory
2) Load ELF binary into memory
3) Make syscall to start PicoRV32 core
4) Syscall uses hypercall to send start interrupt
5) Send start interrupt
0
2
Userspace program
1
3
Linux kernel syscalls
4
Hypervisor hypercall
5

Figure 2: Overview of the process used to start a binary on a PicoRV32 core

We connected the PicoRV32 core to the OpenPiton cache system, creating a tile with a PicoRV32 core instead of an OpenPiton core (Figure 1). This was done by inserting a module that translated memory operations issued by the PicoRV32 core into OpenPiton cache operations.

The resulting design allows the two cores to share memory, and the PicoRV32 core uses the P-Mesh cache coherence protocol without modification. The PicoRV32 core can also receive interprocessor interrupts from the OpenPiton core and is brought out of reset using an interrupt from the OpenPiton core.

We chose PicoRV32 core for several reasons. First, it is open-source and is written in synthesizable Verilog. It has also been applied in a number of settings by the community and has been the subject of formal verification [9]. Second, its simpler microarchitecture meant we would have a heterogeneous ISA, heterogeneous microarchitecture system. We chose a core with a simpler microarchitecture rather than a more complex out-of-order core since this was our first time integrating a new core into OpenPiton.

We expect that integrating a more complex core would follow the same process of inserting a module between the core and the rest of the cache system that translates from the core's interface to the OpenPiton L1.5 cache interface.

Tile Type	Core LUTs	Tile LUTs	Core BRAMs
OpenPiton	36756	64695	24
Pico	1076	21862	0

Table 2: Resource utilization for OpenPiton and PicoRV32 cores and tiles

To test our system, we implemented a way to offload RISC-V binaries to the PicoRV32 core. We wrote a userspace loader program and added two new Linux syscalls and a new SPARC hypercall. The program first allocates a region of memory for the PicoRV32 core to run in. It then loads the RISC-V binary into the allocated memory, writes the start PC to a known address, and sends an interrupt to the PicoRV32 core to bring it out of reset. This process is also shown in Figure 2.

The loader program also proxies any syscalls from the PicoRV32 core using a shared piece of memory. Binaries that run on the PicoRV32 core are linked against a version of Newlib where the syscall stubs are modified to write the syscall number and arguments out to the shared piece of memory. The OpenPiton loader program polls on this memory and when it sees the PicoRV32 core needs a syscall serviced, it reads the number and arguments from memory and makes the syscall in Linux itself. It then writes the result back to the shared memory region, where the PicoRV32 core can read it.

We host the PicoRV32 core since it does not implement the RISC-V privileged spec, but a core with a privileged spec implementation could also be hosted using the same setup. Additionally, system software support for heterogeneous-ISA systems with self-hosting cores in a shared memory system is an active area of research.

For all evaluations we used Xilinx Vivado to synthesize designs for a Digilent Genesys2 FPGA. We compared FPGA resource usage between an OpenPiton tile and a PicoRV32 tile. These results are shown in Table 2. The PicoRV32 core is much smaller than the OpenPiton core, using approximately $\frac{1}{30}th$ the logic resources on FPGA. The resulting PicoRV32 tile uses half the resources of a normal OpenPiton tile. This area reduction is due to the much simpler core design.

We compared the performance of the OpenPiton core and PicoRV32 on 3 microbenchmarks to assess the resource/performance trade-off. One simulated solving the Towers of Hanoi puzzle (`hanoi`), one was a binary search program (`binsearch`), and the third was a quicksort program (`quicksort`).

The PicoRV32 core runs the microbenchmarks more slowly as expected given its more simplistic design. The slowdowns for different benchmarks are shown in Figure 3. `hanoi` and `quicksort` both saw an 8x slowdown. `binsearch` only has a 4x slowdown, because its working set of 10,000 4-byte integers does not fit in the 8KB L1.5 cache. The working set does fit within the L2 although there is still the possibility of conflict misses. As a result of the working set size, both cores are forced to access the L2 or memory often. Since operations that must go to the L1.5 or beyond take approximately the same amount of time for the PicoRV32 core and the OpenPiton core, `binsearch` is less impacted by running on the PicoRV32 core. Considering that the PicoRV32 core is 30 times smaller than the OpenPiton core, a 4 - 8x slowdown seems reasonable, and workloads that spend more time waiting on I/O or memory will be less affected by running on the smaller core. Additionally, we would

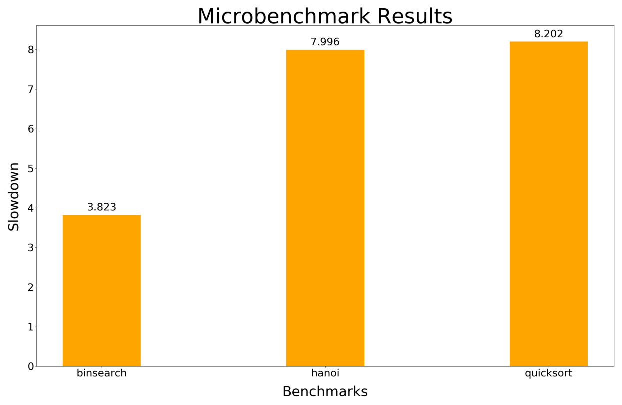

Figure 3: Slowdown for running the microbenchmarks on the PicoRV32 core compared to the OpenPiton core

expect the power consumption of a PicoRV32 tile to be considerably lower.

ACKNOWLEDGEMENTS

This material is based on research sponsored by the NSF under Grant No. CNS-1823222, Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement No. FA8650-18-2-7846 and FA8650-18-2-7852. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA), the NSF, or the U.S. Government.

REFERENCES

[1] Apple. [n. d.]. iPhone 8 and iPhone 8 Plus: A new generation of iPhone . https://www.apple.com/newsroom/2017/09/iphone-8-and-iphone-8-plus-a-new-generation-of-iphone/
[2] ARM. [n. d.]. ARM Technologies: big.LITTLE. https://developer.arm.com/technologies/big-little
[3] Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen, Yanqi Zhou, Alexey Lavrov, Mohammad Shahrad, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff. 2016. OpenPiton: An Open Source Manycore Research Framework. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16)*. ACM, New York, NY, USA, 217–232. https://doi.org/10.1145/2872362.2872414
[4] Sharath K. Bhat, Ajithchandra Saya, Hemedra K. Rawat, Antonio Barbalace, and Binoy Ravindran. 2015. Harnessing Energy Efficiency of heterogeneous-ISA Platforms. In *Proceedings of the Workshop on Power-Aware Computing and Systems (HotPower '15)*. ACM, New York, NY, USA, 6–10. https://doi.org/10.1145/2818613.2818747
[5] Matthew DeVuyst, Ashish Venkat, and Dean M. Tullsen. 2012. Execution Migration in a heterogeneous-ISA Chip Multiprocessor. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVII)*. ACM, New York, NY, USA, 261–272. https://doi.org/10.1145/2150976.2151004
[6] Robert Lyerly, Antonio Barbalace, Christopher Jelesnianski, Vincent Legout, Anthony Carno, and Binoy Ravindran. 2016. Operating System Process and Thread Migration in Heterogeneous Platforms. In *2016 Workshop on Multicore and Rack-Scale systems*.
[7] A. Venkat and D. M. Tullsen. 2014. Harnessing ISA diversity: Design of a heterogeneous-ISA chip multiprocessor. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. 121–132. https://doi.org/10.1109/ISCA.2014.6853218
[8] Clifford Wolf. [n. d.]. PicoRV32. https://github.com/cliffordwolf/picorv32
[9] Clifford Wolf. [n. d.]. PicoRV32 Formal Verification. https://github.com/cliffordwolf/riscv-formal/tree/master/cores/picorv32