OpenPiton: An Open Source Hardware Platform For Your Research

[Extended Abstract]

Jonathan Balkind
Princeton University
jbalkind@princeton.edu

Michael McKeown
Princeton University
mmckeown@princeton.edu

Yaosheng Fu†
NVIDIA
yfu@nvidia.com

Tri Nguyen
Princeton University
trin@princeton.edu

Yanqi Zhou†
Baidu SVAIL
zhouyanqi@baidu.com

Alexey Lavrov
Princeton University
alavrov@princeton.edu

Mohammad Shahrad
Princeton University
mshahrad@princeton.edu

Adi Fuchs
Princeton University
adif@princeton.edu

Samuel Payne†
NVIDIA
spayne@nvidia.com

Xiaohua Liang†
Microsoft
xialian@microsoft.com

Matthew Matl†
University of California, Berkeley
mmatl@eecs.berkeley.edu

David Wentzlaff
Princeton University
wentzlaf@princeton.edu

ABSTRACT

Industry is building larger, more complex, manycore processors on the back of strong institutional knowledge, but academic projects face difficulties in replicating that scale. To alleviate these difficulties and to develop and share knowledge, the community needs open architecture frameworks for simulation, chip design, and software exploration which support extensibility, scalability, and configurability, alongside an established base of verification tools and supported software. In this highlight, we present OpenPiton, an open source framework for building scalable architecture research prototypes from 1 core to 500 million cores. OpenPiton is the world's first open source, general-purpose, multithreaded manycore processor and framework. OpenPiton is highly configurable, providing a rich design space spanning a variety of hardware parameters that researchers can change. OpenPiton designs can be emulated on FPGAs, where they can run full-stack multiuser Debian Linux. OpenPiton is designed to scale to very large core fabrics, enabling researchers to measure operating system, compiler, and software scalability. The mature codebase reflects the complexity of an industrial-grade design and provides the necessary scripts to build new chips, making OpenPiton a natural choice for computer-aided design research. OpenPiton has been validated with a 25-core chip prototype, named Piton, and is bolstered by a validation suite that has thousands of tests, providing an environment to test new hardware designs while verifying the correctness of the whole system. OpenPiton is being actively used in research both internally to Princeton and in the wider community, as well as being adopted in education, industry, and government settings.

1. INTRODUCTION

Building processors for academic research purposes can be a risky proposition. Particularly as processors have grown in size, and with the focus on multicore and manycore processors [17, 19, 20, 21, 14, 22, 6], the number of potential points of failure in chip fabrication has increased drastically. To combat this, the community needs well-tested, open-source, scalable frameworks that they can rely on as baselines to work from and compare against. To reduce "academic time-to-publication", these frameworks must provide robust software tools, mature full-system software stacks, rely on industry-standard languages, and provide thorough test suites. Additionally, to support research in a broad variety of fields, these frameworks must be highly configurable, be synthesisable to FPGA and ASIC for prototyping purposes, and provide the basis for others to tape-out (manufacture) their own, modified academic chips. Building and supporting such an infrastructure is a major undertaking which has prevented such prior designs. Our framework, OpenPiton, attacks this challenge and provides all of these features and more.

OpenPiton is the world's first open source, general-purpose, multithreaded manycore processor. OpenPiton is scalable and portable; the architecture supports addressing for up to 500-million cores, supports shared memory both within a chip and across multiple chips, and has been designed to easily enable high performance 1000+ core microprocessors and beyond. The design is implemented in

The original version of this paper is entitled "OpenPiton: An Open Source Manycore Research Framework" and was published in Proceedings of ASPLOS 2016, Atlanta, GA, April 2-6, 2016, ACM.

† Work was done at Princeton University.

 https://doi.org/10.1145/3366343.

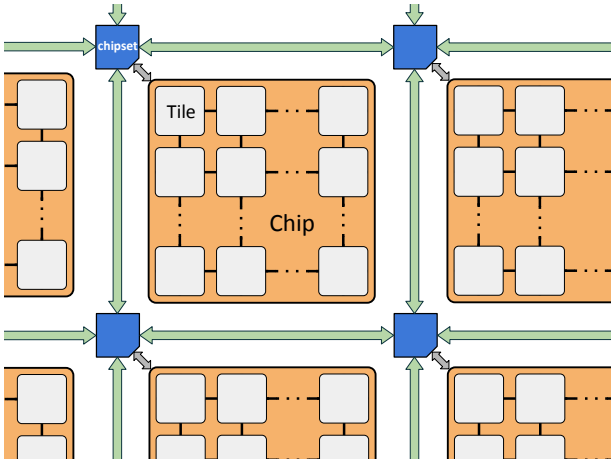

Figure 1: OpenPiton Architecture. Multiple manycore chips are connected together with chipset logic and networks to build large scalable manycore systems. OpenPiton's cache coherence protocol extends off chip.

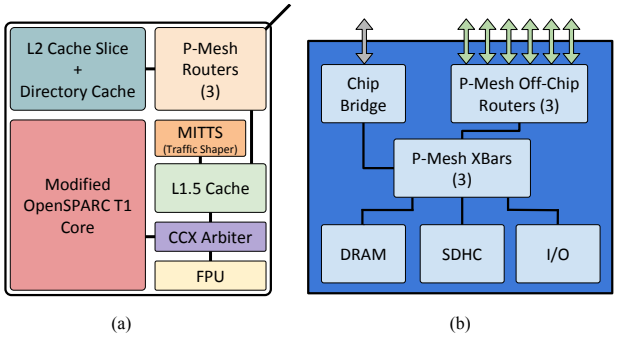

Figure 2: Architecture of (a) a tile and (b) chipset.

Component	Configurability Options
Cores (per chip)	Up to 65,536
Cores (per system)	Up to 500 million
Threads per Core	1/**2**/4
Floating-Point Unit	**Present**/Absent
Stream-Processing Unit	Present/**Absent**
TLBs	8/**16**/32/64 entries
L1 I-Cache	8*/**16**/32KB
L1 D-Cache	4*/**8**/16KB
L1.5 Cache	Number of Sets, Ways (**8KB, 4-way**)
L2 Cache (per tile)	Number of Sets, Ways (**64KB, 4-way**)
Intra-chip Topologies	**2D Mesh**, Crossbar
Inter-chip Topologies	2D Mesh, 3D Mesh, Crossbar, Butterfly Network
Bootloading	SD/SDHC Card, UART

Table 1: Supported OpenPiton Configuration Options. Bold indicates default values. (*Associativity reduced to 2-ways at smallest size)

industry-standard Verilog HDL and does not require the use of any new languages. OpenPiton enables research from the small to the large with demonstrated implementations from the slimmed-down, single-core PicoPiton, which is emulated on a $160 Xilinx Artix 7 at 29.5MHz, up to the 25-core Piton processor which targeted a 1GHz operating point and was recently validated and thoroughly characterised [12, 13].

The OpenPiton platform shown in Figure 1 is a modern, tiled, manycore design consisting of a 64-bit architecture using the mature SPARC v9 ISA with P-Mesh: our scalable cache coherence protocol and network on chip (NoC). OpenPiton builds upon the industry-hardened, open-source OpenSPARC T1 [15, 1, 18] core, but sports a completely scratch-built uncore (caches, cache-coherence protocol, NoCs, NoC-based I/O bridges, etc), a new and modern simulation framework, configurable and portable FPGA scripts, a complete set of scripts enabling synthesis and implementation of ready-to-manufacture chips, and full-stack multiuser Debian Linux support. OpenPiton is available for download at `http://www.openpiton.org`.

OpenPiton has been designed as a platform to enable at-scale research. An explicit design goal of OpenPiton is that it should be easy to use by other researchers. To support this, OpenPiton provides a high degree of integration and configurability as shown in Table 1. Unlike many other designs where the pieces are provided, but it is up to the user to compose them together, OpenPiton is designed with all of the components integrated into the same, easy-to-use, build infrastructure providing **push-button scalability.** Researchers can easily deploy OpenPiton's source code, add in modifications and explore their novel research ideas in the setting of a fully working system. Thousands of targeted, high-coverage test cases are provided to enable researchers to innovate with a safety net that ensures functionality is maintained. OpenPiton's open source nature also makes it easy to release modifications and reproduce previous work for comparison or reuse.

Rather than simply being a platform designed by computer architects for use by computer architects, OpenPiton enables researchers in other fields including operating systems (OS), security, compilers, runtime tools, systems, and computer aided design (CAD) tools to conduct research at-scale. In order to enable such a wide range of applications, OpenPiton is configurable and extensible. The number of cores, attached I/O, size of caches, in-core parameters, and network topology are all configurable from a command-line option or configuration file. OpenPiton is easy to extend; the presence of a well documented core, a well documented coherence protocol, and an easy-to-interface NoC make adding research features straightforward. Research extensions to OpenPiton that have already been built include several novel memory system explorations, an Oblivious RAM controller, and a new in-core thread scheduler. The validated and mature ISA and software ecosystem support OS and compiler research. The release of OpenPiton's scripts for FPGA emulation and chip manufacture make it easy for others to port to new FPGAs or semiconductor process technologies. In particular, this enables CAD researchers who need large netlists to evaluate their algorithms at-scale.

2. THE OPENPITON PLATFORM

OpenPiton is a tiled, manycore architecture, as shown in Figure 1. It is designed to be scalable, both intra-chip and inter-chip, using the P-Mesh cache coherence system.

Intra-chip, tiles are connected via three P-Mesh networks on-chip (NoCs) in a scalable 2D mesh topology (by default). The NoC router address space supports scaling up to 256 tiles in each dimension within a single OpenPiton chip (64K cores/chip).

For inter-chip communication, the chip bridge extends the three NoCs off-chip, connecting the tile array (through the tile in the upper-left) to off-chip logic (chipset). The chipset

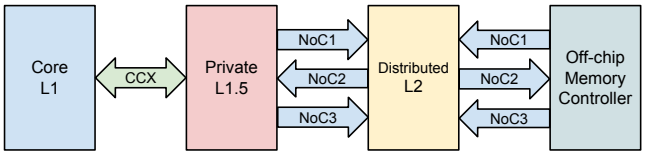

Figure 3: OpenPiton's memory hierarchy datapath.

may be implemented on an FPGA, as a standalone chip, or integrated into an OpenPiton chip.

The extension of the P-Mesh NoCs off-chip allows the seamless connection of multiple OpenPiton chips to create a larger system, as shown in Figure 1. OpenPiton's cache-coherence extends off-chip as well, enabling shared-memory across multiple chips, for the study of even larger shared-memory manycore systems.

2.1 Tile

The architecture of a tile is shown in Figure 2a. A tile consists of a core, an L1.5 cache, an L2 cache, a floating-point unit (FPU), a CPU-Cache Crossbar (CCX) arbiter, a Memory Inter-arrival Time Traffic Shaper (MITTS), and three P-Mesh NoC routers.

The L2 and L1.5 caches connect directly to all three NoC routers, and all messages entering and leaving the tile traverse these interfaces. The CPU Cache-Crossbar (CCX) is the crossbar interface used in the OpenSPARC T1 to connect the cores, L2 cache, FPU, I/O, etc. [1]. In OpenPiton, the L1.5 and FPU are connected to the core by CCX.

2.2 Core

OpenPiton uses the open-source OpenSPARC T1 [15] core with modifications. This core was chosen because of its industry-hardened design, multi-threaded capability, simplicity, and modest silicon area requirements. Equally important, the OpenSPARC framework has a stable code base, implements a mature ISA with compiler and OS support, and comes with a large test suite.

In the default configuration for OpenPiton, as used in Piton, the number of threads is reduced from four to two and the stream processing unit (SPU) is removed from the core to save area. The default Translation Lookaside Buffer (TLB) size is 16 entries but can be increased to 32 or 64, or decreased down to 8 entries.

Additional configuration registers were added to enable extensibility within the core. They are useful for adding additional functionality to the core which can be configured from software, e.g. enabling/disabling functionality, configuring different modes of operation, etc.

2.3 Cache Hierarchy

OpenPiton's cache hierarchy is composed of three cache levels. Each tile in OpenPiton contains private L1 and L1.5 caches and a slice of the distributed, shared L2 cache. The data path of the cache hierarchy is shown in Figure 3.

The memory subsystem maintains cache coherence using our coherence protocol, called P-Mesh. It adheres to the memory consistency model used by the OpenSPARC T1. Coherent messages between L1.5 caches and L2 caches communicate through three NoCs, carefully designed to ensure deadlock-free operation.

2.3.1 L1 Caches

The L1 caches are reused from the OpenSPARC T1 design with extensions for configurability. They are composed of separate L1 instruction and L1 data caches, both of which are write-through and 4-way set-associative. By default, the L1 data cache is an 8KB cache and its line size is 16-bytes. The 16KB L1 instruction cache has a 32-byte line size.

2.3.2 L1.5 Data Cache

The L1.5 (comparable to L2 caches in other processors) both transduces the OpenSPARC T1's CPU-Cache Crossbar (CCX) protocol to P-Mesh's NoC coherence packet formats, and acts as a write-back layer, caching stores from the write-through L1 data cache. Its parameters match the L1 data cache by default.

The L1.5 communicates to and from the core through the CCX bus, preserved from the OpenSPARC T1. When a memory request results in a miss, the L1.5 translates and forwards it to the L2 through the network-on-chip (NoC) channels. Generally, the L1.5 issues requests on NoC1, receives data on NoC2, and writes back modified cache lines on NoC3, as shown in Figure 3.

The L1.5 is inclusive of the L1 data cache; each can be independently sized with independent eviction policies. For space and performance, the L1.5 does not cache instructions– these cache lines are bypassed directly to the L2 cache.

2.3.3 L2 Cache

The L2 cache (comparable to a last-level L3 cache in other processors) is a distributed, write-back cache shared by all tiles. The default cache configuration is 64KB per tile and 4-way set associativity, but both the cache size and associativity are configurable. The cache line size is 64 bytes, larger than the line sizes of caches lower in the hierarchy. The integrated directory cache has 64 bits per entry, so it can precisely keep track of up to 64 sharers by default.

The L2 cache is inclusive of the private caches (L1 and L1.5). Cache line way mapping between the L1.5 and the L2 is independent and is entirely subject to the replacement policy of each cache. Since the L2 is distributed, cache lines consecutively mapped in the L1.5 are likely to be distributed across multiple L2 tiles (L2 tile referring to a portion of the distributed L2 cache in a single tile).

The L2 is the point of coherence for all cacheable memory requests. All cacheable memory operations (including atomic operations such as compare-and-swap) are ordered, and the L2 strictly follows this order when servicing requests. The L2 also keeps the instruction and data caches coherent, per the OpenSPARC T1's original design. When a line is present in a core's L1 instruction cache and is loaded as data, the L2 sends invalidations to the relevant instruction caches before servicing the load.

2.4 P-Mesh Network On-chip

There are three P-Mesh NoCs in an OpenPiton chip. The NoCs provide communication between the tiles for cache coherence, I/O, memory traffic, and inter-core interrupts. They also route traffic destined for off-chip to the chip bridge. The packet format contains 29 bits of core addressability, making it scalable up to 500 million cores.

To ensure deadlock-free operation, the L1.5 cache, L2 cache, and memory controller give different priorities to different NoC channels; NoC3 has the highest priority, next is NoC2, and NoC1 has the lowest priority. Thus, NoC3 will never be blocked. In addition, all hardware components are

designed such that consuming a high priority packet is never dependent on lower priority traffic.

Classes of coherence operations are mapped to NoCs based on the following rules, as depicted in Figure 3:

- NoC1 messages are initiated by requests from the private cache (L1.5) to the shared cache (L2).
- NoC2 messages are initiated by the shared cache (L2) to the private cache (L1.5) or memory controller.
- NoC3 messages are responses from the private cache (L1.5) or memory controller to the shared cache (L2).

2.5 Chipset

The chipset, shown in Figure 2b, houses the I/O, DRAM controllers, chip bridge, P-Mesh chipset crossbar, and P-Mesh inter-chip network routers. The chip bridge de-multiplexes traffic from the attached chip back into the three physical NoCs. The traffic then passes through a Packet Filter (not shown), which modifies packet destination addresses based on the memory address in the request and the set of devices on the chipset. The chipset crossbar (a modified network router) then routes the packets to their correct destination device. If the traffic is not destined for this chipset, it is passed to the inter-chip network routers, which route the traffic to another chipset according to the inter-chip routing protocol. Traffic destined for the attached chip is directed back through similar paths to the chip bridge.

Inter-chip Routing.

The inter-chip network router is configurable in terms of router degree, routing algorithm, buffer size, etc. This enables flexible exploration of different router configurations and network topologies. Currently, we have implemented and verified crossbar, 2D mesh, 3D mesh, and butterfly networks. Customized topologies can be explored by reconfiguring the network routers.

2.6 Configurability

OpenPiton was designed to be a configurable platform, making it useful for many applications. Table 1 shows OpenPiton's configurability options, highlighting the large design space that it offers.

2.6.1 PyHP for Verilog

In order to provide low effort configurability of our Verilog RTL, we make use of a Python pre-processor, the Python Hypertext Processor (PyHP) [16]. PyHP was originally designed for Python dynamic webpage generation and is akin to PHP. We have adapted it for use with Verilog code. Parameters can be passed into PyHP, and arbitrary Python code can be used to generate testbenches or modules. PyHP enables extensive configurability beyond what is possible with Verilog generate statements alone.

2.6.2 Core and Cache Configurability

OpenPiton's core configurability parameters are shown in Table 1. The default parameters are shown in bold. OpenPiton preserves the OpenSPARC T1's ability to modify TLB sizes (from 8 to 64, in powers of two), thread counts (from 1 to 4), and the presence or absence of the FPU and SPU. Additionally, OpenPiton's L1 data and instruction caches can be doubled or halved in size (associativity drops to 2 when reducing size).

Leveraging PyHP, OpenPiton provides parameterizable memories for simulation or FPGA emulation. In addition, custom or proprietary memories can easily be used for chip development. This parameterization enables the configurability of cache parameters. The size and associativity of the L1.5 and L2 caches are configurable, though the line size remains static.

2.6.3 Manycore Scalability

PyHP also enables the creation of scalable meshes of cores, drastically reducing the code size and complexity in some areas adopted from the original OpenSPARC T1. OpenPiton automatically generates all core instances and wires for connecting them from a single template instance. This reduces code complexity, improves readability, saves time when modifying the design, and makes the creation of large meshes straightforward. The creation of large two-dimensional mesh interconnects of up to 256x256 tiles is reduced to a single instantiation. The mesh can be any rectangular configuration, and the dimensions do not need to be powers of two. This was a necessary feature for the 5x5 (25-core) Piton processor.

2.6.4 NoC Topology Configurability

P-Mesh provides other NoC connection topologies than the default two-dimensional mesh used in OpenPiton. The coherence protocol only requires that messages are delivered in-order from one point to another point. Since there are no inter-node ordering requirements, the NoC can easily be swapped out for a crossbar, higher dimension router, or higher radix design. Our configurable P-Mesh router can be reconfigured to a number of topologies shown in Table 1. For intra-chip use, OpenPiton can be configured to use a crossbar, which has been tested with four and eight cores with no test regressions. Other NoC research prototypes can easily be integrated and their performance, energy, and other characteristics can be determined through RTL, gate-level simulation, or FPGA emulation.

2.6.5 Chipset Configurability

The P-Mesh chipset crossbar is configurable in the number of ports to connect the myriad devices OpenPiton users may have. There is a single XML file where the chipset devices and their address ranges are specified, so connecting a new device needs only a Verilog instantiation and an XML entry. PyHP is used to automatically connect the necessary P-Mesh NoC connections and Packet Filters.

We have so far connected a variety of devices through P-Mesh on the chipset. These include DRAM, Ethernet, UART, SD, SDHC, VGA, PS/2 keyboards, and even the MIAOW open source GPU [2]. These devices are driven by the OpenPiton core and perform their own DMA where necessary, routed over the chipset crossbar.

2.6.6 Multi-chip Scalability

Similar to the on-chip mesh, PyHP enables the generation of a network of chips starting with the instantiation of a single chip. OpenPiton provides an address space for up to 8192 chips, with 65,536 cores per chip. By using the scalable P-Mesh cache coherence mechanism built into OpenPiton, half-billion core systems can be built. This configurability enables the building of large systems to test ideas at scale.

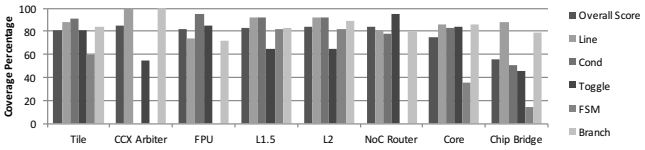

Figure 4: Test suite coverage results by module (default OpenPiton configuration).

3. VALIDATION

3.1 Platform Stability

One of the benefits of OpenPiton is its stability, maturity, and active support. Much of this is inherited from the OpenSPARC T1 core, which has a stable code base and has been studied for years, allowing the code to be reviewed and bugs fixed by many people. In addition, it implements a mature, commercial, and open ISA, SPARC V9. This means that there is existing full tool chain support for OpenPiton, including Debian Linux OS support, a compiler, and an assembler. SPARC is supported on a number of OSs including Debian Linux, Oracle's Linux for SPARC, and OpenSolaris (and its successors). Porting the OpenSPARC T1 hypervisor required changes to fewer than 10 instructions, and a newer Debian Linux distribution was modified with open source, readily available, OpenSPARC T1-specific patches written as part of Lockbox [3, 4].

OpenPiton provides additional stability on top of what is inherited from OpenSPARC T1. The tool flow was updated to modern tools and ported to modern Xilinx FPGAs. OpenPiton is also used extensively for research internal to Princeton. This means there is active support for OpenPiton, and the code is constantly being improved and optimized, with regular releases over the last several years. In addition, the open sourcing of OpenPiton has strengthened its stability as a community has built.

3.1.1 Validation

When designing large scale processors, simulation of the hardware design is a must. OpenPiton supports one open source and multiple commercial Verilog simulators, which can simulate the OpenPiton design at rates up to tens or hundreds of kilohertz. OpenPiton inherited and then extended the OpenSPARC T1's large test suite with thousands of directed assembly tests, randomized assembly test generators, and tests written in C. This includes tests for not only the core, but the memory system, I/O, cache coherence protocol, etc. Additionally, the extensions like Execution Drafting (Section 4.1.1) have their own test suites. When making research modifications to OpenPiton, the researcher can rely on an established test suite to ensure that their modifications did not introduce any regressions. In addition, the OpenPiton documentation details how to add new tests to validate modifications and extend the existing test suite. Researchers can also use our scripts to run large regressions in parallel (to tackle the slower individual execution), automatically produce pass/fail reports and coverage reports (as shown in Figure 4), and run synthesis to verify that synthesis-safe Verilog has been used. Our scripts support the widely-used SLURM job scheduler and integrate with Jenkins for continuous integration testing.

Linux for SPARC is hosted at `https://oss.oracle.com/projects/linux-sparc/`

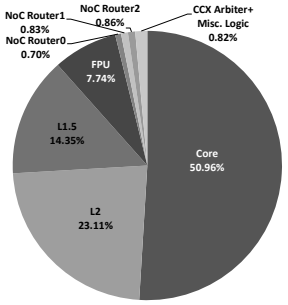

Figure 5: Tile area breakdown for FPGA PicoPiton

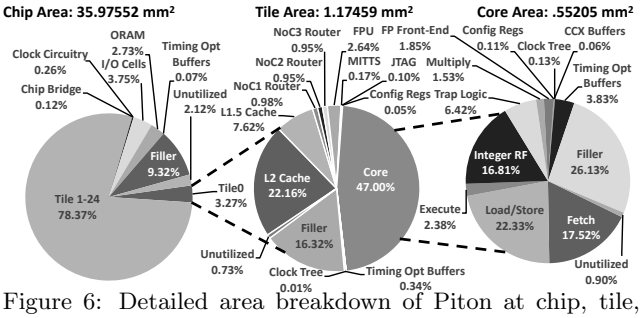

Figure 6: Detailed area breakdown of Piton at chip, tile, and core levels. Reproduced from [13].

3.2 FPGA Prototyping

OpenPiton can also be emulated on FPGA, which provides the opportunity to prototype the design, emulated at tens of megahertz, to improve throughput when running our test suite or more complex code, such as an interactive operating system. OpenPiton is actively supported on three Xilinx FPGA platforms: Artix-7 (Digilent Nexys Video), Kintex-7 (Digilent Genesys 2) and Virtex-7 (VC707 Evaluation Board). An external port is also maintained for the Zynq-7000 (ZC706 Evaluation Board). Figure 5 shows the area breakdown for a minimized "PicoPiton" core, implemented for an Artix-7 FPGA (Digilent Nexys 4 DDR).

OpenPiton designs have the same features as the Piton processor, validating the feasibility of that particular design (multicore functionality, etc.), and can include the chip bridge to connect multiple FPGAs via an FPGA Mezzanine Card (FMC) link. All of the FPGA prototypes feature a full system (chip plus chipset), using the same codebase as the chipset used to test the Piton processor.

OpenPiton on FPGA can load bare-metal programs over a serial port and can boot full stack multiuser Debian Linux from an SD/SDHC card. Booting Debian on the Genesys2 board running at 87.5MHz takes less than 4 minutes (and booting to a bash shell takes just one minute), compared to 45 minutes for the original OpenSPARC T1, which relied on a tethered MicroBlaze for its memory and I/O requests. This boot time improvement combined with our push-button FPGA synthesis and implementation scripts drastically increases productivity when testing operating system or hardware modifications.

3.3 The Princeton Piton Processor

The Piton processor prototype [12, 13] was manufactured in March 2015 on IBM's 32nm SOI process with a target clock frequency of 1GHz. It features 25 tiles in a 5x5 mesh on a 6mm x 6mm ($36mm^2$) die. Each tile is two-way threaded

and includes three research projects: ExecD [11], CDR [8], and MITTS [23], while an ORAM [7] controller was included at the chip level. The Piton processor provides validation of OpenPiton as a research platform and shows that ideas can be taken from inception to silicon with OpenPiton.

With Piton, we also produced the first detailed power and energy characterisation of an open source manycore design implemented in silicon [13]. This included characterising energy per instruction, NoC energy, voltage versus frequency scaling, thermal characterisation, and memory system energy, among other properties. All of this was done in our lab, running on the Piton processor with the OpenPiton chipset implemented on FPGA. Performing such a characterisation yielded new insights into the balance between recomputation and data movement, the energy cost of differing operand values, and a confirmation of earlier results [9] that showed that NoCs do not dominate manycore processors' power consumption. Our study also produced what we believe is the most detailed area breakdown of an open source manycore, which we reproduce in Figure 6. All characterisation data from our study, as well as designs for the chip printed circuit board (PCB), are now open source at `http://www.openpiton.org`.

3.4 Synthesis and Back-end Support

OpenPiton provides scripts to aid in synthesis and back-end physical design for generating realistic area results or for manufacturing new chips based on OpenPiton. The scripts are identical to the ones used to tape-out the Piton processor, however the scripts have been made process agnostic and references to the specific technology used have been removed due to proprietary foundry intellectual property concerns. Directions are included with OpenPiton which describe how to port to a new foundry kit. This allows the user to download OpenPiton, link to the necessary process development kit files, and run our full tool flow to produce the chip layout for a new instance of OpenPiton. In this sense, OpenPiton is portable across process technologies and provides a complete ecosystem to implement, test, prototype, and tape-out (manufacture) research chips.

4. APPLICATIONS

Table 2 presents a taxonomy of open source processors which highlights imporant parameters for research. Since OpenPiton's first release in 2015, it has been used across a wide range of applications and research domains, some of which are described in this section.

4.1 Internal Research Case Studies

4.1.1 Execution Drafting

Execution Drafting [11] (ExecD) is an energy saving microarchitectural technique for multithreaded processors which leverages duplicate computation. ExecD takes over the thread selection decision from the OpenSPARC T1 thread selection policy and instruments the front-end to achieve energy savings. ExecD required modifications to the OpenSPARC T1 core and thus was not as simple as plugging a standalone module into the OpenPiton system. The core microarchitecture needed to be understood, and the implementation tightly integrated with the core. Implementing ExecD in OpenPiton revealed several implementation details that had been abstracted away in simulation, such as tricky divergence conditions in the thread synchronization mechanisms. This reiterates the importance of taking research designs to implementation in an infrastructure like OpenPiton.

ExecD must be enabled by an ExecD-aware operating system. Our public Linux kernel and OpenPiton hypervisor repositories contain patches intended to add support for ExecD. These patches were developed as part of a single-semester undergraduate OS research project.

4.1.2 Coherence Domain Restriction

Coherence Domain Restriction [8] (CDR) is a novel cache coherence framework designed to enable large scale shared memory with low storage and energy overhead. CDR restricts cache coherence of an application or page to a subset of cores, rather than keeping global coherence over potentially millions of cores. In order to implement it in OpenPiton, the TLB is extended with extra fields and both the L1.5 and L2 cache are modified to fit CDR into the existing cache coherence protocol. CDR is specifically designed for large scale shared memory systems such as OpenPiton. In fact, OpenPiton's million-core scalability is not feasible without CDR because of increasing directory storage overhead.

4.1.3 MITTS

The Memory Inter-arrival Time Traffic Shaper [23] (MITTS) enables a manycore system or an IaaS cloud system to provision memory bandwidth in the form of a memory request inter-arrival time distribution at a per-core or per-application basis. A runtime system configures MITTS knobs in order to optimize different metrics (e.g., throughput, fairness). MITTS sits at the egress of the L1.5 cache, monitoring the memory requests and stalling the L1.5 when it uses bandwidth outside its allocated distribution. MITTS has been integrated with OpenPiton and works on a per-core granularity, though it could be easily modified to operate per-thread.

MITTS must also be supported by the OS. Our public Linux kernel and OpenPiton hypervisor repositories contain patches for supporting the MITTS hardware. With these patches, developed as an undergraduate thesis project, Linux processes can be assigned memory inter-arrival time distributions, as they would in an IaaS environment where the customer paid for a particular distribution corresponding with their application's behaviour. The OS configures the MITTS bins to correspond with each process's allocated distribution, and MITTS enforces the distribution accordingly.

4.2 External Research Use

A number of external researchers have already made considerable use of OpenPiton. In a CAD context, Lerner et al. [10] present a development workflow for improving processor lifetime, based on OpenPiton and the gem5 simulator, which is able to improve the design's reliability time by 4.1x.

OpenPiton has also been used in a security context as a testbed for hardware trojan detection. OpenPiton's FPGA emulation enabled Elnaggar et al. [5] to boot full-stack Debian Linux and extract performance counter information while running SPEC benchmarks. This project moved quickly from adopting OpenPiton to an accepted publication in a matter of months, thanks in part to the full-stack OpenPiton system that can be emulated on FPGA.

Processor	Architecture	FPU	OS	MMU	HW Multithreaded	Multicore/ Manycore/GPU	Prototype Core Count	NoC	HDL	Back-end Scripts	License
pAVR	8b AVR	✗	✗	✗	✗	No	-	✗	VHDL	✗	GPL v2
openMSP430	16bMSP430	✗	✓(RTOS)	✗	✗	No	-	✗	Verilog	✗	BSD
CPU86	16b x86	✗	✓	✗	✗	No	-	✗	VHDL	✗	GPL
Zet	16b x86	✗	✓	✗	✗	No	-	✗	Verilog	✗	GPL v3
LatticeMico32	32b LatticeMico32	✗	✓	✗	✗	No	-	✗	Verilog	✗	GPL
ZPU	32b MIPS	✗	✓	✗	✗	No	-	✗	VHDL	✗	FreeBSD & GPL
SecretBlaze	32b MicroBlaze	✗	✗	✗	✗	No	-	✗	VHDL	✗	GPL v3
AltOr32	32b ORBIS	✗	✓	✗	✗	No	-	✗	Verilog	✗	LGPL v3
aeMB	32b MicroBlaze	✗	✓	✗	✓	No	-	✗	Verilog	✗	LGPL v3
Amber	32b ARM v2a	✗	✓	✗	✗	No	-	✗	Verilog	✗	LGPL
OpenRISC	32b/64b ORBIS	✓	✓	✓	✗	No	-	✗	Verilog	✗	LGPL
MIPS32 r1	32b MIPS32 r1	✗	✓	✗	✓	No	-	✗	Verilog	✗	LGPL v3
LEON 3	32b SPARC V8	✓($)	✓	✓	✗	SMP/AMP	-	✗	VHDL	✗	GPL
OpenScale	32b MicroBlaze	✗	✓(RTOS)	✗	✗	Manycore	FPGA/6	✓	VHDL	✗	GPL v3
XUM	32b MIPS32 r2	✗	✓	✗	✓	Manycore	FPGA/8	✓	Verilog	✗	LGPL v3
PicoRV32	32b RISC-V	✗	✗	✗	✗	No	FPGA/1	✗	Verilog	✗	ISC
PULP-RI5CY	32b RISC-V	✓	✓(RTOS)	✗	✗	Manycore	Chip/9	✗	SystemVerilog	✗	Solderpad 0.51
PULP-Zeroriscy	32b RISC-V	✗	✓(RTOS)	✗	✗	Multicore	Chip/1	✗	SystemVerilog	✗	Solderpad 0.51
Nyuzi GPGPU	Nyami ISA	✓	✓	✓	✓	GPGPU	FPGA	✓	SystemVerilog	✗	Apache 2.0
MIAOW GPGPU	AMD Southern Islands	✓	✗	✗	✓	GPU	FPGA/1	✓	Verilog	✗	BSD 3-Clause
OpTiMSoC	32b/64b ORBIS	✓	✓	✓	✗	Manycore	FPGA/4	✓	SystemVerilog	✗	MIT
Simply RISC S1	64b SPARC V9	✓	✓	✓	✗	No	-	✗	Verilog	✗	GPL v2
BERI	64b MIPS/CHERI	✓	✓	✓	✓(BERI2)	Multicore	FPGA/4	✗	Bluespec	✗	BERI HW-SW
OpenSPARC T1/T2	64b SPARC V9	✓	✓	✓	✓	Multicore	Chip/8	✗	Verilog	✗	GPL v2
Rocket	64b RISC-V	✓	✓	✓	✗	Manycore	Chip/8	✓	Chisel	✗	BSD 3-Clause
AnyCore	64b RISC-V	✗	✗	✗	✗	No	Chip/1	✗	SystemVerilog	✗	BSD 3-Clause
PULP-Ariane	64b RISC-V	✗	✓	✗	✗	Manycore	Chip/1	✗	SystemVerilog	✗	Solderpad 0.51
BOOM	64b RISC-V	✓	✓	✓	✗	Manycore	FPGA	✓	Chisel	✗	BSD 3-Clause
OpenPiton	64b SPARC V9	✓	✓	✓	✓	Manycore	Chip/25	✓	Verilog	✓	BSD 3-Clause & GPL v2

Table 2: Taxonomy of differences of open source processors (table data last checked in April 2018).

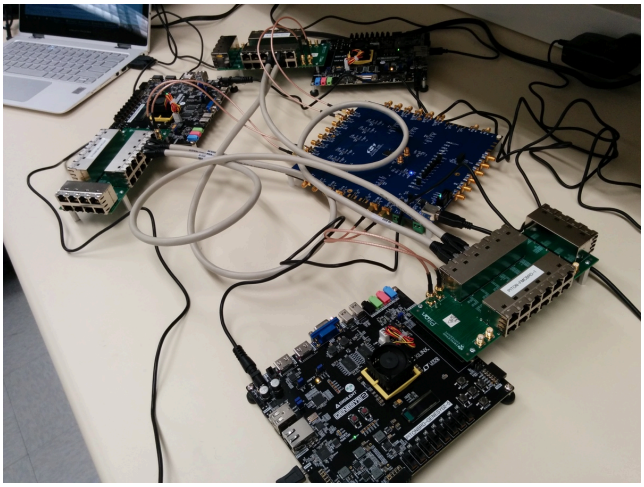

Figure 7: Three OpenPiton FPGAs connected by 9 gigabit per second serial P-Mesh links.

Oblivious RAM (ORAM) [7] is a memory controller designed to eliminate memory side channels. An ORAM controller was integrated into the 25-core Piton processor, providing the opportunity for secure access to off-chip DRAM. The controller was directly connected to OpenPiton's NoC, making the integration straightforward. It only required a handful of files to wrap an existing ORAM implementation, and once it was connected, its integration was verified in simulation using the OpenPiton test suite.

4.3 Educational Use

We have been using OpenPiton in coursework at Princeton, in particular our senior undergraduate Computer Architecture and graduate Parallel Computation classes. A few of the resulting student projects are described here.

Core Replacement.

Internally, we have tested replacements for the OpenSPARC T1 core with two other open source cores. These modifications replaced the CCX interface to the L1.5 cache with shims which translate to the L1.5's interface signals. These shims require very little logic but provide the cores with fully cache-coherent memory access through P-Mesh. We are using these cores to investigate manycore processors with heterogeneous ISAs.

Multichip Network Topology Exploration.

A senior undergraduate thesis project investigated the impact of interchip network topologies for large manycore processors. Figure 7 shows multiple FPGAs connected over a high-speed serial interface, carrying standard P-Mesh packets at 9 gigabits per second. The student developed a configurable P-Mesh router for this project which is now integrated as a standard OpenPiton component.

MIAOW.

A student project integrated the MIAOW open source GPU [2] with OpenPiton. An OpenPiton core and a MIAOW core can both fit onto a VC707 FPGA with the OpenPiton core acting as a host, in place of the Microblaze that was used in the original MIAOW release. The students added MIAOW to the chipset crossbar with a single entry in its XML configuration. Once they implemented a native P-Mesh interface to replace the original AXI-Lite interface, MIAOW could directly access its data and instructions from memory without the core's assistance.

Hardware Transactional Memory.

Another student project was the implementation of a Hard-

ware Transactional Memory system in OpenPiton. The students learned about the P-Mesh cache coherence protocol from the OpenPiton documentation, before modifying it, including adding extra states to the L1.5 cache, and producing a highly functional prototype in only six weeks. The OpenPiton test suite was central to verifying that existing functionality was maintained in the process.

Cache Replacement Policies.

A number of student groups have modified the cache replacement policies of both the L1.5 and L2 caches. OpenPiton enabled them to investigate the performance and area tradeoffs of their replacement policies across multiple cache sizes and associativities in the context of a full-stack system, capable of running complex applications.

4.4 Industrial and Governmental Use

So far we are aware of multiple CAD vendors making use of OpenPiton internally for testing and educational purposes. These users provide extra confidence that the RTL written for OpenPiton will be well supported by industrial CAD tools, as vendors often lack large scale designs to validate the functionality of their tools. In government use, DARPA has identified OpenPiton as a benchmark for use in the POSH program.

5. FUTURE

OpenPiton has a bright future. It not only has active support from researchers at Princeton, but has a vibrant external user base and development community. The OpenPiton team has run four tutorials at major conferences and numerous tutorials at interested universities and will continue to run more tutorials. The future roadmap for OpenPiton includes adding additional configurability, support for more FPGA platforms and vendors, the ability to emulate in the Cloud by using Amazon AWS F1 instances, more core types plugged into the OpenPiton infrastructure, and integration with other emerging open source hardware projects. OpenPiton has demonstrated the ability to enable research at hardware speeds, at scale, and across different areas of computing research. OpenPiton and other emerging open source hardware projects have the potential to have significant impact not only on how we conduct research and educate students, but also design chips for commercial and governmental applications.

6. ACKNOWLEDGMENTS

This material is based on research sponsored by the NSF under Grants No. CNS-1823222, CCF-1217553, CCF-1453112, and CCF-1438980, AFOSR under Grant No. FA9550-14-1-0148, Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement No. FA8650-18-2-7846 and FA8650-18-2-7852 and DARPA under Grants No. N66001-14-1-4040 and HR0011-13-2-0005. 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, AFOSR, DARPA, or the U.S. Government. We thank Paul Jackson, Ting-Jung Chang, Ang Li, Fei Gao, Katie Lim, Felix Madutsa, and Kathleen Feng for their important contributions to OpenPiton.

7. REFERENCES

[1] *OpenSPARC T1 Microarchitecture Specification.* Santa Clara, CA, 2006.

[2] R. Balasubramanian, V. Gangadhar, Z. Guo, C.-H. Ho, C. Joseph, J. Menon, M. P. Drumond, R. Paul, S. Prasad, P. Valathol, and K. Sankaralingam. Enabling gpgpu low-level hardware explorations with miaow: An open-source rtl implementation of a gpgpu. *ACM Trans. Archit. Code Optim.*, 12(2), June 2015.

[3] D. Bittman, D. Capelis, and D. Long. Introducing seaos. In *Information Science and Applications (ICISA), 2014 International Conference on*, pages 1–3, May 2014.

[4] D. J. Capelis. Lockbox: Helping computers keep your secrets. Technical Report UCSC-WASP-15-02, University of California, Santa Cruz, Nov. 2015.

[5] R. Elnaggar, K. Chakrabarty, and M. B. Tahoori. Run-time hardware trojan detection using performance counters. In *2017 IEEE International Test Conference (ITC)*, pages 1–10, Oct 2017.

[6] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, pages 365–376, New York, NY, USA, 2011. ACM.

[7] C. W. Fletcher, L. Ren, A. Kwon, M. van Dijk, and S. Devadas. Freecursive oram: [nearly] free recursion and integrity verification for position-based oblivious ram. In *Proceedings of the Twentieth Int. Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, pages 103–116, New York, NY, USA, 2015. ACM.

[8] Y. Fu, T. M. Nguyen, and D. Wentzlaff. Coherence domain restriction on large scale systems. In *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, pages 686–698, New York, NY, USA, 2015. ACM.

[9] J. S. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff. Energy characterization of a tiled architecture processor with on-chip networks. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, ISLPED '03, pages 424–427, New York, NY, USA, 2003. ACM.

[10] S. Lerner and B. Taskin. Workload-aware asic flow for lifetime improvement of multi-core iot processors. In *2017 18th International Symposium on Quality Electronic Design (ISQED)*, pages 379–384, March 2017.

[11] M. McKeown, J. Balkind, and D. Wentzlaff. Execution drafting: Energy efficiency through computation deduplication. In *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pages 432–444, Dec 2014.

[12] M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, J. Balkind, A. Lavrov, M. Shahrad, S. Payne, and D. Wentzlaff. Piton: A manycore processor for multitenant clouds.

IEEE Micro, 37(2):70–80, Mar 2017.

[13] M. McKeown, A. Lavrov, M. Shahrad, P. Jackson, Y. Fu, J. Balkind, T. Nguyen, K. Lim, Y. Zhou, and D. Wentzlaff. Power and energy characterization of an open source 25-core manycore processor. In *High Performance Computer Architecture (HPCA), IEEE Int. Symposium on*, 2018.

[14] B. Miller, D. Brasili, T. Kiszely, R. Kuhn, R. Mehrotra, M. Salvi, M. Kulkarni, A. Varadharajan, S.-H. Yin, W. Lin, A. Hughes, B. Stysiack, V. Kandadi, I. Pragaspathi, D. Hartman, D. Carlson, V. Yalala, T. Xanthopoulos, S. Meninger, E. Crain, M. Spaeth, A. Aina, S. Balasubramanian, J. Vulih, P. Tiwary, D. Lin, R. Kessler, B. Fishbein, and A. Jain. A 32-core risc microprocessor with network accelerators, power management and testability features. In *IEEE Int. Solid-State Circuits Conf. Digest of Tech. Papers*, pages 58–60, Feb 2012.

[15] Oracle. OpenSPARC T1. `http://www.oracle.com/technetwork/systems/opensparc/opensparc-t1-page-1444609.html`.

[16] PyHP. PyHP Official Home Page. `http://pyhp.sourceforge.net`.

[17] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: A many-core x86 architecture for visual computing. *ACM Trans. Graph.*, 27(3):18:1–18:15, Aug. 2008.

[18] J. Szefer, W. Zhang, Y.-Y. Chen, D. Champagne, K. Chan, W. Li, R. Cheung, and R. Lee. Rapid single-chip secure processor prototyping on the opensparc fpga platform. In *Rapid System Prototyping (RSP), 2011 22nd IEEE International Symposium on*, pages 38–44, May 2011.

[19] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, et al. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *Solid-State Circuits, IEEE Journal of*, 43(1):29–41, 2008.

[20] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. Brown III, and A. Agarwal. On-chip interconnection architecture of the Tile Processor. *IEEE Micro*, 27(5):15–31, Sept. 2007.

[21] D. Wentzlaff, C. J. Jackson, P. Griffin, and A. Agarwal. Configurable fine-grain protection for multicore processor virtualization. In *Proceedings of the Annual Int. Symp. on Computer Architecture*, pages 464–475, Washington, DC, USA, 2012.

[22] D. H. Woo and H.-H. S. Lee. Extending amdahl's law for energy-efficient computing in the many-core era. *Computer*, (12):24–31, 2008.

[23] Y. Zhou and D. Wentzlaff. Mitts: Memory inter-arrival time traffic shaping. In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, pages 532–544, Piscataway, NJ, USA, 2016. IEEE Press.