OpenPiton: An Open Source Manycore Research Framework

Jonathan Balkind Michael McKeown Yaosheng Fu Tri Nguyen
Yanqi Zhou Alexey Lavrov Mohammad Shahrad Adi Fuchs
Samuel Payne * Xiaohua Liang Matthew Matl David Wentzlaff

Princeton University

{jbalkind,mmckeown,yfu,trin,yanqiz,alavrov,mshahrad,adif}@princeton.edu,
spayne@nvidia.com, {xiaohua,mmatl,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, synthesis, and software exploration which support extensibility, scalability, and configurability, alongside an established base of verification tools and supported software. In this paper 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 leverages the industry hardened OpenSPARC T1 core with modifications and builds upon it with a scratch-built, scalable uncore creating a flexible, modern manycore design. In addition, OpenPiton provides synthesis and backend scripts for ASIC and FPGA to enable other researchers to bring their designs to implementation. OpenPiton provides a complete verification infrastructure of over 8000 tests, is supported by mature software tools, runs full-stack multiuser Debian Linux, and is written in industry standard Verilog. Multiple implementations of OpenPiton have been created including a taped-out 25-core implementation in IBM's 32nm process and multiple Xilinx FPGA prototypes.

1. Introduction

Multicore and manycore processors have been growing in size and popularity fueled by the growth in single-chip transistor count and the need for higher performance. This trend has been widespread across the industry with manycore processors being fabricated by numerous vendors [49, 70, 82, 84, 86]. Academia has also heavily studied multicore and manycore [28, 47, 87], though often through the use of software simulation [14, 21, 31, 50, 59, 81] and modeling [43, 72] tools. While software tools are convenient to work with and provide great flexibility, they do not provide the high performance, characterization fidelity (power, area, and frequency), and ability to perform research at-scale that taking a manycore design to RTL, FPGA, or silicon implementation can provide.

A major barrier limiting manycore research is the lack of an easy to use, highly scalable, open source, manycore processor design. In order to maximize benefit, such a design and framework would have robust software tools, be easily configurable, provide a mature tool ecosystem with OS support, be written in an industry standard HDL, be easily synthesizable to FPGA and ASIC, and provide a large test suite. Building and supporting such an infrastructure is a major undertaking which has prevented such prior designs. Our

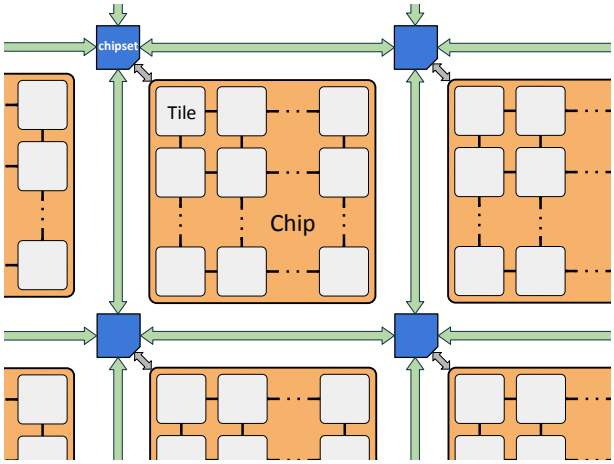


Figure 1: OpenPiton Architecture. Multiple manycore chips are connected together with chipset logic and networks to build large scalable manycore systems. OpenPiton's cache coherence protocol extends off chip.

* Now at Nvidia

ASPLOS '16, April 02 - 06, 2016, Atlanta, GA, USA

DOI: http://dx.doi.org/10.1145/2872362.2872414

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. 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 a distributed, directory-based, cache coherence protocol (shared distributed L2) implemented across three physical, 2D mesh Networks-on-Chip (NoCs). OpenPiton contains a pipelined dual-precision FPU per core and supports native multithreading to hide memory latency. OpenPiton builds upon the industry hardened OpenSPARC T1 [6, 58, 78] core, but sports a completely scratch-built uncore (caches, cache-coherence protocol, NoCs, scalable interrupt controller, NoC-based I/O bridges, etc), a new and modern simulation and synthesis framework, a modern set of FPGA scripts, a complete set of ASIC back-end scripts enabling chip tape-out, and full-stack multiuser Debian Linux support. OpenPiton is available for download at `http://www.openpiton.org`.

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 **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 a single lite-core, PicoPiton, on a Xilinx Artix 7 operating at 29.5MHz up to Piton, our 25-core custom ASIC implementation of OpenPiton, taped-out at 1GHz in 32nm silicon.

OpenPiton has been designed as a platform to enable at-scale manycore 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. 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.** Computer architecture 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. Over 8000 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.

Beyond being a platform designed by computer architects for use by computer architects, OpenPiton enables researchers in other fields including OS, security, compilers, runtime tools, systems, and even CAD tool design to conduct research at-scale. In order to enable such a wide range

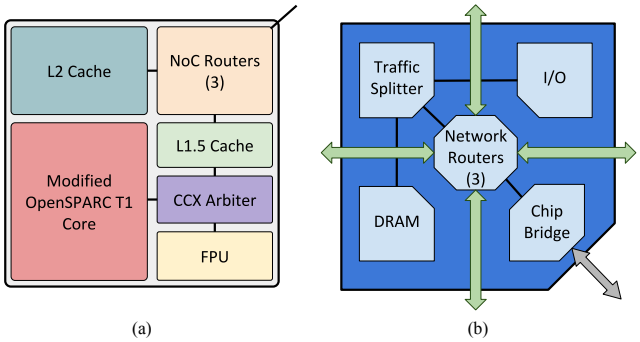


Figure 2: Architecture of (a) a tile and (b) chipset.

of applications, OpenPiton is configurable and extensible. The number of cores, attached I/O, size of caches, number of TLB entries, presence of an FPU, number of threads, and network topology are all configurable from a single 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 easy. 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 ASIC synthesis, FPGA synthesis, and back-end (Place and Route) scripts make it easy for others to port to new process technologies or FPGAs. In particular, this enables CAD researchers who need large netlists to evaluate their algorithms at-scale.

The following are our key contributions:

- Creation and release of the world's first open source, general-purpose, multithreaded manycore processor.
- Detailed description of the OpenPiton architecture including a novel coherent memory system that seamlessly integrates on-chip and off-chip networks.
- Presentation of multiple use-cases of configurability and extensibility of the OpenPiton Architecture.
- Comparison of FPGA and ASIC implementations of OpenPiton
- Characterization of backend synthesis runtimes.
- Characterization of the test coverage of the OpenPiton test suite.
- Qualitative comparison against prior open source processors.

2. Architecture

OpenPiton is a tiled-manycore architecture, as shown in Figure 1. It is designed to be scalable, both intra-chip and inter-chip.

Intra-chip, tiles are connected via three networks on-chip (NoCs) in a 2D mesh topology. The scalable tiled architecture and mesh topology allow for the number of tiles within an OpenPiton chip to be configurable. In the default config-

uration, the NoC router address space supports scaling up to 256 tiles in each dimension within a single OpenPiton chip (64K cores/chip).

Inter-chip, an off-chip interface, known as the chip bridge, connects the tile array (through the tile in the upper-left) to off-chip logic (chipset). The chipset may be implemented on an FPGA, ASIC, or integrated on an OpenPiton chip. The chip bridge extends the three NoCs off-chip, multiplexing them over a single link.

The extension of NoCs off-chip allows the seamless connection of multiple OpenPiton chips to create a larger system, as illustrated in Figure 1. The cache-coherence protocol extends off-chip as well, enabling shared-memory across multiple chips. A core in one chip may be cache coherent with cores in another chip. This enables 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 modified OpenSPARC T1 [58] core, an L1.5 cache, an L2 cache, a floating-point unit (FPU), a CCX arbiter, and three 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. [6]. The L1.5 is responsible for transducing between CCX and the NoC router protocol. The CCX arbiter de-multiplexes memory and floating-point requests from the core to the L1.5 cache and FPU, respectively, and arbitrates responses back to the core.

2.2 Core

OpenPiton uses the OpenSPARC T1 [58] core with minimal 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 taped out OpenPiton core, the default configuration for OpenPiton, the number of threads is reduced from four to two. This was primarily done to reduce the area requirement of the core and to reduce the pressure on the memory system. By default, the stream processing unit (SPU), essentially a cryptographic functional unit, is also removed from the core to save area. Instructions intending to utilize the SPU will trigger a trap and are emulated in software. The default TLB size is 16 entries to reduce area (which reflects the reduction in the number of threads), but it can be increased to 64 or decreased down to 8 entries.

Additional configuration registers were added to enable extensibility within the core. They are implemented as memory-mapped registers in an alternate address space

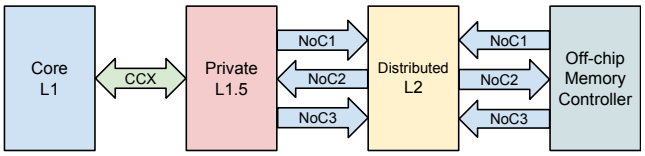


Figure 3: OpenPiton's memory hierarchy datapath.

(one way to implement CPU control registers in SPARC). These configuration registers can be 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.

Many of the modifications to the OpenSPARC T1 core, like the ones above, have been made in a configurable way. Thus, it is possible to configure the core to the original OpenSPARC T1 specifications or a set of parameters different from the original OpenSPARC T1 as is the default OpenPiton core. More details on configurability are discussed in Section 3.1.

2.3 Cache Hierarchy

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

2.3.1 L1 Cache

The L1 cache is reused, with minimal modifications, from the OpenSPARC T1 design. It is tightly integrated to the OpenSPARC T1 pipeline, and composed of two separate caches: the L1 data cache and L1 instruction cache. The L1 data cache is an 8KB write-through cache; it is 4-way set-associative and the line size is 16-bytes. The 16KB L1 instruction cache is similarly 4-way set associative but with a 32-byte line size. Both L1 caches' sizes can be configured, as detailed in Section 3.1.2.

There are two issues with the original OpenSPARC T1 L1 cache design which made it suboptimal for use in a scalable multicore and hence required changes in OpenPiton. First, write-through caches require extremely high write-bandwidth to the next cache level, which is likely to overwhelm and congest NoCs in manycore processors with distributed caches. This necessitates a local write-back cache. Second, the cache communication interface needs to be compliant with OpenPiton's cache coherence protocol, i.e., tracking MESI cache line states, honoring remote invalidations, and communicating through OpenPiton's NoCs instead of the OpenSPARC T1's crossbar. Rather than modifying the existing RTL for the L1s, we introduced an extra cache level (L1.5) to tackle both issues.

2.3.2 L1.5 Data Cache

The L1.5 serves as both the glue logic, transducing the OpenSPARC T1's crossbar protocol to OpenPiton's NoC

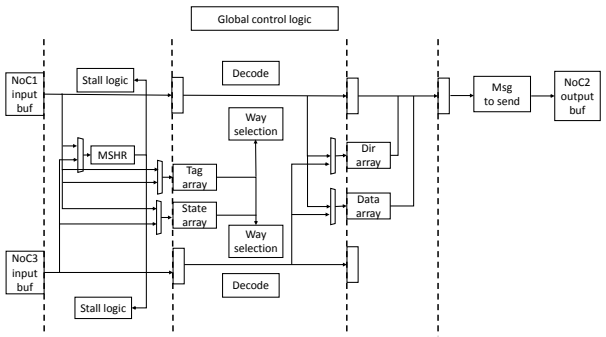


Figure 4: The architecture of the L2 cache.

coherence packet formats, and a write-back layer, caching stores from the write-through L1 data cache. It is an 8KB 4-way set associative write-back cache (the same size as the L1 data cache by default) with configurable associativity and size. The line size is the same as the L1 data cache at 16-bytes.

The L1.5 communicates requests and responses to and from the core through CCX. The CCX bus is preserved as the primary interface to the OpenSPARC T1. The L1.5 CCX interface could relatively easily be replaced with other interfaces like AMBA or AXI to accommodate different cores. When a memory request results in a miss, the L1.5 translates and forwards request 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.

While the L1.5 was named as such during the development of the Piton ASIC prototype, in traditional computer architecture contexts it would be appropriate to call it the "private L2" and to call the next level cache the "shared/distributed L3". The L1.5 is inclusive of the L1 data cache; each can be independently sized with independent eviction policies. As a space- and performance-conscious optimization, the L1.5 does not cache instructions–these cache lines are bypassed directly between the L1 instruction cache and the L2. It is possible to modify the L1.5 to also cache instructions.

2.3.3 L2 Cache

The L2 cache 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 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. In fact, since the L2 is distributed, cache lines consecutively mapped in the L1.5 are likely to be strewn across multiple L2 tiles (L2 tile referring to a portion of the distributed L2 cache in a single tile). By default, OpenPiton maps cache lines using constant strides with the lower address bits across all L2 tiles, but Coherence Domain Restriction (CDR) [30], an experimental research feature integrated into OpenPiton, can be used to interleave cache lines belonging to a single application or page across a software-specified set of L2 tiles.

As shown in Figure 4, the L2 cache is designed with dual parallel pipelines. The first pipeline (top) receives cache miss request packets from lower in the cache hierarchy on NoC1 and sends memory request packets to off-chip DRAM and cache fill response packets to lower in the cache hierarchy on NoC2. The second pipeline (bottom) receives memory response packets from off-chip DRAM and modified cache line writeback packets from lower in the cache hierarchy on NoC3. The first L2 pipeline contains 4 stages and the second pipeline contains only 3 stages since it does not transmit output packets. The interaction between the L2 and the three NoCs is also depicted in Figure 3.

2.4 Cache Coherence and Memory Consistency Model

The memory subsystem maintains cache coherence with a directory-based MESI coherence protocol. It adheres to the TSO 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.

The L2 is the point of coherence for all memory requests, except for non-cacheable loads and stores which directly bypass the L2 cache. All other 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, coherence between the two L1 caches is maintained at the L2. When a line is present in a core's L1 instruction cache and is loaded as data, the L2 will send invalidations to the relevant instruction caches before servicing the load.

High-level features of the coherence protocol include:

- 4-step message communication
- Silent eviction in Exclusive and Shared states
- No acknowledgments for dirty write-backs
- Three 64-bit physical NoCs with point-to-point ordering
- Co-location of L2 cache and coherence directory

2.5 Interconnect

There are two major interconnection types used in OpenPiton, the NoCs and the chip bridge.

2.5.1 Network On-chip (NoC)

There are three NoCs in an OpenPiton chip. The NoCs connect tiles in a 2D mesh topology. The main purpose of the NoCs is to provide communication between the tiles for cache coherence, I/O and memory traffic, and inter-core interrupts. They also route traffic destined for off-chip to the chip bridge. The NoCs maintain point-to-point ordering between a single source and destination, a feature often leveraged to maintain TSO consistency. In a multi-chip configuration, OpenPiton uses similar configurable NoC routers to route traffic between chips.

The three NoCs are physical networks (no virtual channels) and each consists of two 64-bit uni-directional links, one in each direction. The links use credit-based flow control. Packets are routed using dimension-ordered wormhole routing to ensure a deadlock-free network. The packet format preserves 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. While the cache coherence protocol is designed to be logically deadlock free, it also depends on the physical layer and routing to also be deadlock free.

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).

2.5.2 Chip Bridge

The chip bridge connects the tile array to the chipset, through the upper-left tile, as shown in Figure 1. All memory and I/O requests are directed through this interface to be served by the chipset. Its main purpose is to transparently multiplex the three physical NoCs over a single, narrower link in pin-limited chip implementations.

The chip bridge contains asynchronous buffers to bridge between I/O and core clock domains. It implements three virtual off-chip channels over a single off-chip physical channel, providing the necessary buffering and arbitration logic. The off-chip channel contains two 32-bit unidirectional links, one in each direction, and uses credit-based flow control. At 350MHz, Piton's target I/O frequency, the chip bridge provides a total bandwidth of 2.8GB/s.

2.6 Chipset

The chipset, shown in Figure 2b, houses the I/O, DRAM controllers, chip bridge, traffic splitter, and inter-chip network routers. The chip bridge brings traffic from the attached chip into the chipset and de-multiplexes it back into the three physical NoCs. The traffic is passed to the inter-chip network routers, which routes it to the traffic splitter if it is destined for this chipset. The traffic splitter multiplexes requests to the DRAM controller or I/O devices, based on the address of the request, to be serviced. If the traffic is not destined for this chipset, it is routed 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.

2.6.1 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.

We have proven that our network routing protocols can safely extend and be further routed (under some constraints) off chip while maintaining their deadlock-free nature.

2.7 Floating-point Unit

We utilize the FPU from OpenSPARC T1 [58]. In OpenPiton, there is a one-to-one relationship between cores and FPUs, in contrast to the OpenSPARC T1, which shares one FPU among eight cores [6]. This was primarily done to boost floating-point performance and to avoid the complexities of having shared FPUs among a variable number of tiles. The CCX arbiter always prioritizes the L1.5 over the FPU in arbitration over the shared CCX interface into the core.

3. Platform Features

This section discusses the flagship features of OpenPiton that make it a compelling platform for doing manycore research.

3.1 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.

3.1.1 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) [63]. 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

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

Table 1: Supported OpenPiton Configuration Options

can be used to generate testbenches or modules. PyHP enables extensive configurability beyond what is possible with Verilog generate statements alone. The motivation for using PyHP came from the inability to view the intermediate code after pre-processing Verilog generate statements, which made debugging difficult.

3.1.2 Core Configurability

As mentioned previously, the cores at the heart of OpenPiton are two-way threaded by default with an FPU per tile. OpenPiton preserves the OpenSPARC T1's ability to modify TLB sizes, thread counts, and the presence or absence of the FPU and SPU. As shown in Table 1, OpenPiton offers the option to select from 1 to 4 hardware threads, vary the TLB between 8 and 64 entries (in powers of two), and choose whether or not to include an FPU, an SPU, or both in each tile. Additionally, the L1 data and instruction caches can be doubled in size. Future support will include configuration of the L1 caches' associativity.

3.1.3 Cache Configurability

Leveraging PyHP, OpenPiton provides parameterizable generic flop-based memories for simulation in addition to the infrastructure for using custom or proprietary SRAMs. This enables the configurability of cache parameters. The associativity and size of the L1.5 and L2 caches are configurable, though the line size remains static. OpenPiton also includes scripts which automate the generation of correctly sized RAMs necessary for FPGA implementations, providing a push-button approach to FPGA implementation.

3.1.4 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 our 5x5 (25-core) tape-out.

3.1.5 NoC Topology Configurability

Two-dimensional mesh is not the only possible NoC connection topology for 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. Currently, OpenPiton can be configured with 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 or gate-level simulation, or by FPGA emulation.

3.1.6 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. In conjunction with the scalable 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.

3.1.7 Case Study: PicoPiton

To demonstrate the elasticity of OpenPiton's design space, we created a minimized configuration called PicoPiton, which fits on a Digilent Nexys 4 DDR board. Currently, this board costs only $160 (US) for academic purchase, making it an ideal solution for low-cost academic prototyping with OpenPiton. The small area of this board required size reductions for many structures. The final version of PicoPiton is single threaded with no FPU, SPU, or chip bridge, 8 TLB entries and a 32KB L2 cache. PicoPiton is able to reach a clock frequency of 29.5MHz and is shown in Figure 5 executing "Hello, world!" over a serial connection.

3.2 Extensibility

One of the most prominent design goals for OpenPiton was to create a framework which could easily be extended. This enables rapid prototyping of research ideas backed by infrastructure for validation, characterization, and implementation. This section discusses some of the ways in which OpenPiton is extensible and presents case studies demonstrating how OpenPiton has already been used in research.

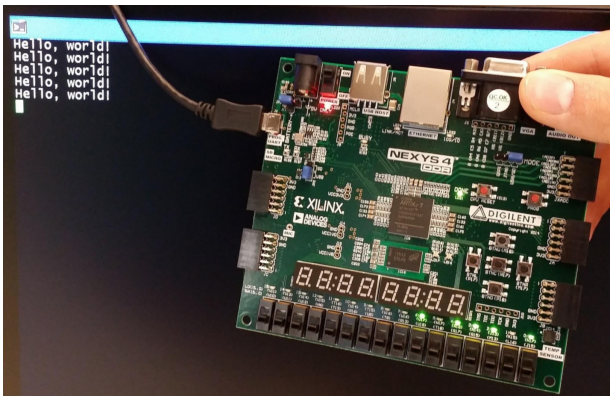


Figure 5: PicoPiton printing Hello World to UART.

3.2.1 Core Replacement

Given their importance [28, 41], support for heterogeneous systems architectures in OpenPiton is compelling. Future plans for OpenPiton include building a heterogeneous mesh of cores and accelerators connected to OpenPiton's NoC and memory system. Integrating an OpenSPARC T2 core would be straightforward as it uses a similar CCX crossbar interface. With the removal of the CCX arbiter, the L1.5 could likely communicate with any other TSO-compliant core. If the core's L1 cache were a write-back cache, a simple transducer could potentially replace the L1.5 cache entirely.

3.2.2 Accelerators

Accelerators can be easily connected to the NoC thereby enabling integration into the cache coherence protocol. As OpenPiton's coherence protocol is open source and documented, this provides a unique opportunity for accelerator researchers. Accelerators which participate in the coherence protocol can work cooperatively and in parallel with other accelerators or general purpose cores, utilizing OpenPiton's modern NoC based architecture and cache-coherence protocol. OpenPiton enables this type of research at scale, only possible previously through high-level simulation.

3.2.3 AXI-Lite Bridge

OpenPiton includes an AXI4-Lite bridge that provides connectivity to a wide range of I/O devices by interfacing memory mapped I/O operations from the NoCs to AXI-Lite. Given OpenPiton prototypes make use of Xilinx FPGAs, Xilinx-provided intellectual property (IP) cores compatible with AXI and AXI-Lite interfaces can be seamlessly integrated with OpenPiton. The Xilinx UART IP core, which uses the AXI-Lite interface, has been successfully integrated into the OpenPiton FPGA prototypes through memory mapped I/O. By using a standard interface like AXI, many I/O devices can be interfaced with OpenPiton and existing drivers can be reused.

3.2.4 Case Studies

Several case studies demonstrate how the OpenPiton platform has thus far been used to enable research.

Execution Drafting Execution Drafting [46] (ExecD) is an energy saving microarchitectural technique for multi-threaded processors which leverages duplicate computation. ExecD is a more ad-hoc case study, as it required modification to the OpenSPARC T1 core. Therefore, it was not as simple as plugging a standalone module into the OpenPiton system. Instead the core microarchitecture needed to be understood and the implementation tightly integrated with the core. However, the integration of ExecD into 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 designs to implementation in an infrastructure like OpenPiton.

ExecD takes over the thread selection decision from the OpenSPARC T1 thread selection policy and instruments the front-end to achieve energy savings. ExecD is a good example of the addition of configuration registers in OpenPiton. It utilizes some of the configuration register address space to set whether ExecD is enabled, which thread synchronization method is used, etc.

Coherence Domain Restriction Coherence Domain Restriction [30] (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.

Oblivious RAM Oblivious RAM (ORAM)[29, 32, 75] is a memory controller designed to eliminate memory side channels. An ORAM controller was integrated into the 25-core Piton ASIC, 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.

3.3 Platform Stability

One of the benefits of OpenPiton is its stability, maturity, and active support. Much of this is inherited from using 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 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 an actively supported platform, with a number of supported OSs and Oracle recently releasing Linux for SPARC in late 2015[1]. Porting the OpenSPARC T1 hypervisor required changes to fewer than 10 instructions, and a newer Debian Linux distribution was modified with readily available, open source, OpenSPARC T1-specific patches written as part of Lockbox[15, 20].

Another feature inherited from using the OpenSPARC T1 core is the large test suite. This includes tests for not only the core, but the memory system, I/O, cache coherence protocol, etc. When making research modifications to OpenPiton, this allows the researcher to 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.

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 internally. This means there is active support for OpenPiton and the code is constantly being improved and optimized. In addition, the open sourcing of OpenPiton will strengthen its stability as a community is built around it.

3.4 Validation

OpenPiton features more than 8000 directed assembly tests for testing not only core functionality, but that of the extensions like ExecD (Section 3.2.4). Validation is done through a mix of assembly tests, randomized assembly test generators, and tests written in C. Scripts are provided that run large regressions in parallel with the SLURM job scheduler and automatically produce pass/fail reports, coverage reports, and even run chip synthesis to verify that synthesis-safe Verilog has been used.

3.4.1 Randomized Testing

The OpenPiton test suite contains a number of randomized assembly test generators written in Perl, known as "PAL" files. These tests have a number of parameters and can be seeded to generate as many test instances as are desired. 2100 of the core OpenPiton tests make use of these PAL test generators, handling behaviors covering memory, branches, ALU, FPU, thread scheduling, and more.

3.4.2 C Tests

While the OpenPiton validation methodology focuses primarily on directed and randomized assembly tests, the infrastructure also supports the use of tests written in C. We have written a small stable of C tests akin to microbenchmarks which can perform any standard C functions (including console I/O but excluding file I/O) and link against static libraries.

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

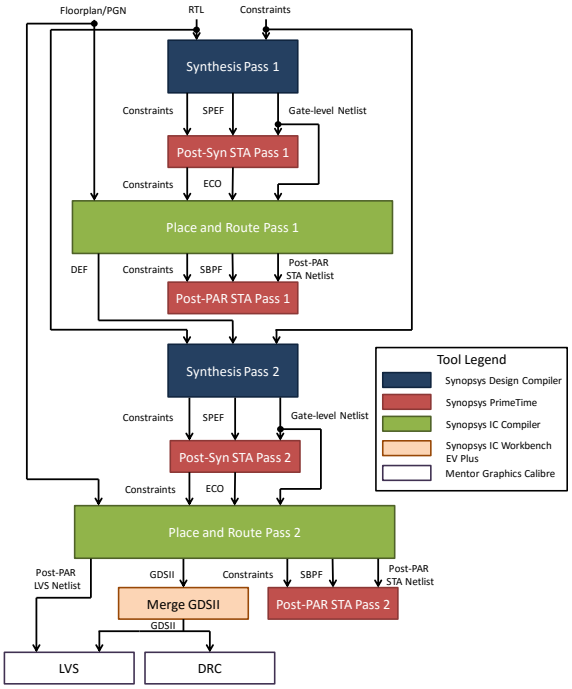


Figure 6: Synthesis and back-end tool flow.

3.4.3 Coverage Reports

In order to gain an understanding of how effective the testing regime is at a given time and to best focus testing efforts, coverage reports are automatically generated alongside the results of a regression. These reports include coverage breakdowns per module and per module instantiation and include line, toggle, condition, branch and finite state machine coverage.

3.5 Synthesis and Back-end Support

OpenPiton provides scripts to aid in synthesis and back-end for generating realistic area results or for taping-out new designs based on OpenPiton. The scripts are identical to the ones used to tape-out the Piton ASIC prototype, however references to the specific technology used have been removed due to proprietary foundry IP concerns and the scripts have been made process agnostic. 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 the full tool flow to tape-out a specific instance of OpenPiton. In this sense, OpenPiton is portable across process technologies and provides a complete ecosystem to implement, test, prototype, and tape-out research.

The tool flow provided with OpenPiton is shown in Figure 6 and is mainly a Synopsys tool flow. The figure shows a two-pass flow, however the number of passes is config-

urable. Increasing the number of passes improves the quality of results, but with diminishing returns. The Verilog RTL along with design constraints are first passed to Synopsys Design Compiler, which synthesizes a gate-level netlist from the behavioral RTL. The resulting netlist, along with the constraints, are passed to Synopsys PrimeTime to perform post-synthesis static timing analysis (STA). This analyzes the gate-level netlist against the constraints specified and reports the results. If the constraints are not met, Synopsys PrimeTime may output an engineering change order (ECO) file, which suggests modifications to the netlist to meet the constraints.

The gate-level netlist, constraints, ECO file, and physical floorplan (specified by the user) are passed to Synopsys IC Compiler to perform placement and routing (PAR). However, before PAR, the ECO modifications are applied to the gate-level netlist to give Synopsys IC Compiler a better chance of meeting the constraints. After PAR is complete, the post-PAR netlist is again passed to Synopsys PrimeTime, along with the constraints, to perform STA and check the design against the constraints. Although not shown in the figure, Synopsys PrimeTime may output an ECO file again, which can be fed back into Synopsys IC compiler, along with the post-PAR netlist and physical layout library to apply the ECO. We have found the ECOs from PrimeTime to be very useful in meeting timing.

The output of this flow is a fully placed and routed design in GDSII format. If it meets the constraints, design rule checking (DRC) and layout versus schematic checking (LVS) are performed. However, it is likely the constraints may not be met after the first pass. In this case, results can be improved by performing the same flow up to this point a second time, while passing physical information from the output of the first IC Compiler pass and the ECO from the output of PrimeTime. Any number of passes through this flow is possible, but we saw diminishing returns after two passes. If constraints are still not met, it may be the case that the constraints and/or floorplan must be modified.

After a GDSII of the design that meets the constraints is obtained, it must be merged with the GDSII of any third-party IP to generate the final GDSII. This is done with the Synopsys IC Workbench EV Plus tool. After the final merged GDSII is generated, it is passed to Mentor Graphics Calibre for LVS and DRC checking. DRC requires the GDSII and the DRC deck from the process development kit. LVS only requires the GDSII and the post-PAR gate-level netlist.

A final consideration for synthesis and back-end flow is what to use for the input RTL. If one were to give the full OpenPiton RTL as input to synthesis, the flow would take an unreasonable amount of time and may not even complete successfully. For this reason, the design is generally broken down into hierarchical blocks. The leaf blocks are run through the flow first, and imported as black-boxes in the flow for modules higher in the hierarchy. This is done until the top-level chip is reached. Since the hierarchy may depend on process technology, design modifications, etc. the OpenPiton synthesis and back-end scripts make it easy to modify and define new module hierarchies.

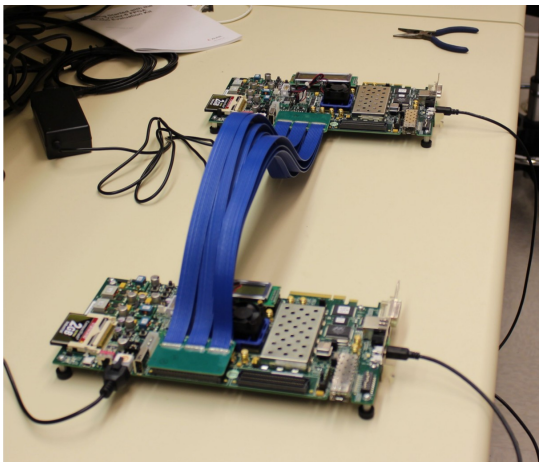


Figure 7: Two Virtex 6 (ML605) boards connected by the chip bridge.

Board	Frequency (1 Core)	Cores
Xilinx VC707 (Virtex-7)	67MHz	4
Digilent Genesys 2 (Kintex-7)	60MHz	2
Digilent Nexys 4 DDR (Artix-7)	29MHz	1
Xilinx ML605 (Virtex-6)	18MHz	2

Table 2: FPGA Boards Currently Supported by OpenPiton

3.5.1 Gate-level Simulation

Gate-level simulation provides more accurate results than behavioral simulation because it is based on post-synthesis netlists and can account for delays through standard cells. OpenPiton extends the gate-level simulation framework from the OpenSPARC T1 and is capable of simulating the entire chip or any sub-modules. Since timing information is available from the standard cells, gate-level simulation is able to check for potential timing violations in the design. The gate-level simulation results can also be compared against the behavioral simulation outputs for functional verification or fed into PrimeTime PX to estimate power consumption. Gate-level power estimation is valuable, as it provides more accurate power estimates than tools that rely on modeling in software simulations.

3.6 Prototyping

This section discusses the prototype implementations of specific OpenPiton instances, which validate it as a viable, modern, and competitive platform.

Figure 8: OpenPiton running on a Xilinx VC707 board executing Tetris on full-stack multiuser Debian Linux

3.6.1 FPGA Prototypes

OpenPiton has been successfully ported to the Virtex-6 (ML605 Evaluation Board), Artix-7 (Digilent Nexys 4 DDR), Kintex-7 (Digilent Genesys 2) and Virtex-7 (VC707 Evaluation Board) Xilinx FPGA platforms, as shown in Table 2, to prototype the design and provide improved test throughput. These prototypes validate the full spectrum of OpenPiton's configurability, from the PicoPiton Artix-7 design at the low-end intended for very small prototypes, to the Virtex-7, which is large enough to fit four OpenPiton cores.

The Virtex and Kintex designs have the same features as the ASIC prototype, validating the feasibility of that particular design (multicore functionality, etc.), and can even include the chip bridge to connect multiple FPGAs via a FPGA Mezzanine Card (FMC) link, as shown in Figure 7. All of the FPGA prototypes feature the NoC to DDR transducer, located on an external FPGA for Piton.

After verifying the functionality of the NoC to AXI-Lite bridge in simulation, it took just a few hours to integrate a serial port and run the first I/O-based "Hello, world!" test on PicoPiton as shown in Figure 5. We expect users of the OpenPiton FPGA prototypes to achieve a similarly quick turnaround from initial Verilog implementation to getting results at speeds far surpassing behavioral simulation.

OpenPiton on the Virtex and Kintex boards can load bare-metal programs over a serial port and can boot full stack multiuser Debian Linux from an SD or MicroSD card, depicted in Figure 8 running Tetris. Booting Debian on the VC707 board running at 67MHz takes about 12 minutes, compared to 45 minutes for the original OpenSPARC T1, which relied on a tethered MicroBlaze for its memory and I/O requests. This improvement combined with the move to Xilinx's Vivado Suite for FPGA synthesis and implementation drastically increased productivity when testing operating system or hardware modifications.

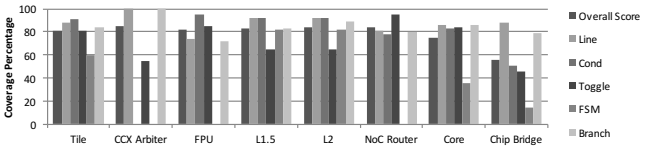


Figure 9: Test suite coverage results by module (default OpenPiton configuration).

3.6.2 ASIC Prototype

The Piton ASIC prototype was taped out 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 including ExecD and CDR, while ORAM was included at the chip level. The ASIC provides validation of OpenPiton as a research platform and shows that ideas can be taken from inception to silicon with OpenPiton.

4. Results

This section provides experimental results from OpenPiton simulation and prototyping.

4.1 Simulation

Verilog simulation in OpenPiton utilizes Synopsys VCS, though there are plans to expand to other Verilog simulators in the future. OpenPiton Verilog simulation runs at 4.85 kilo-instructions per second (KIPS) per tile with Synopsys VCS on a 2.4GHz Intel Westmere Xeon E7-4870 processor. While simulation speed varies with tile count, test stimuli, etc., this value is obtained from averaging across the OpenPiton test suite, which includes various numbers of cores and instruction mixes, and is normalized to a single tile. While the simulation speed may seem slow compared to many architectural simulators, it is a very detailed RTL simulation. In addition, the ability to synthesize OpenPiton to an FPGA makes up for this shortcoming.

Another important metric of OpenPiton is the design coverage resulting from executing the full test suite. This provides an idea of how well the test suite exercises the design. In addition, running coverage after making modifications can give insight into how well the modifications are tested. Figure 9 shows the different types of coverage (line, condition, toggle, FSM, branch) resulting from running the full OpenPiton test suite for all major architectural blocks.

Each architectural block includes the coverage of all blocks beneath it in the module hierarchy. For example, the tile coverage results include the weighted average of coverage results from the CCX arbiter, FPU, L1.5, L2, NoC router, and OpenSPARC T1 core. The missing bars in the figure are not due to zero coverage, but due to the coverage tool's inability to identify that type of coverage for that particular

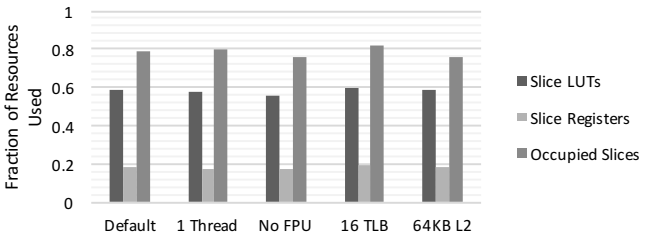


Figure 10: Virtex 6 area comparison for different configurations.

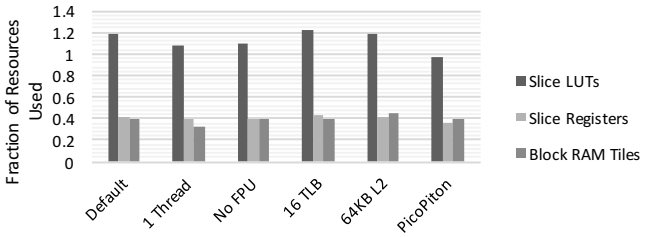


Figure 11: Artix 7 area comparison for different configurations.

design. For example, the CCX arbiter does not have a bar for condition coverage, as there are no complex conditional statements in that design. The interconnections also see low coverage as some paths are not thoroughly used (e.g. the chip bridge sees little NOC1 traffic and NOC2/NOC3 traffic is largely unidirectional when cross-chip sharing is minimal). Overall, the coverage is good, which demonstrates that the OpenPiton test suite does a good job of exercising the OpenPiton design.

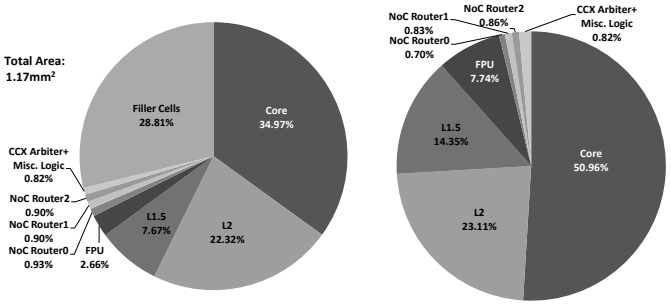
4.2 Prototyping

Our FPGA prototyping efforts have so far resulted in prototypes for four different Xilinx boards: Virtex 6 (ML605), Artix 7 (Nexys 4 DDR), Kintex 7 (Genesys 2), and Virtex 7 (VC707). These prototypes are capable of achieving clock frequencies of 18MHz, 29.5MHz, 60MHz, and 67MHz respectively, as summarized in Table 2.

The default configuration discussed here has two threads, 8 entry TLBs, 8KB L1D, 16KB L1I, 8KB L1.5, 32KB L2, and the FPU included. The area breakdowns for the default configuration on each board are listed in Table 3. Note that the default configuration is too large for the Artix 7 and had to be pared down to fit. Figures 10 and 11 show the resource utilization for different configuration options on the Virtex 6 (post-place and route) and Artix 7 (post-synthesis) FPGAs, as a proportion of the available resources on each FPGA. These figures indicate the rough cost of modifying a particular configuration value, but have some counter-intuitive results. For example, increasing the L2 to 64KB on the Virtex 6 causes Xilinx ISE to better map the design to included BRAM, thus decreasing the amount of area used.

	Slice LUTs	Slice Regs	Occupied Slices
Virtex 6 240T	88,682	56,141	29,691
(Available)	150,720	301,440	37,680
	Slice LUTs	Slice Regs	Block RAM Tiles
Artix 7 100T	75,253	52,993	53
(Available)	63,400	126,800	135
	Slice LUTs	Slice Regs	Block RAM Tiles
Kintex 7 325T	68,976	50403	54
(Available)	203,800	407,600	445
	Slice LUTs	Slice Regs	Block RAM Tiles
Virtex 7 485T	68,303	51,084	23
(Available)	303,600	607,200	1,030

Table 3: Default resource utilization vs available resources for each FPGA.



(a) ASIC Piton (b) FPGA PicoPiton

Figure 12: Tile area breakdown.

Submodule	SYN	PAR	STA	ECO	DRC	LVS	Total
NoC Router	0.32	2.35	0.02	N/A	0.04	0.04	2.80
L1.5	0.40	15.45	0.05	3.55	0.24	0.12	34.18
Core	1.16	36.82	0.19	3.51	1.09	0.32	78.04
Tile	1.11	22.95	0.12	2.69	2.10	0.49	55.89
Chip	4.20	79.19	0.67	11.36	9.04	0.93	104.91

Table 5: Time durations (in hours) of selected synthesis and back-end stages for selected submodules. If any stage executes more than once (STA or for multi-pass flow), the maximum duration is shown.

The area breakdown of a default single tile on Artix 7 is shown in Figure 12b. This breakdown is based on Xilinx Vivado "cells", rather than LUT or BRAM utilization.

4.3 ASIC Prototype

The Piton ASIC prototype was taped-out on the IBM 32nm SOI process with a target clock frequency of 1GHz. The area breakdown of a single tile is shown in Figure 12a. The total tile area is 1.17mm^2 with about 30% of the area taken up by filler cells. Space was overprovisioned by about 30% (70% utilization) in floorplanning stages to give PAR a better chance of meeting the constraints. This space was filled with filler cells after the design was successfully PARed.

Measuring with a 2.4GHz Intel Westmere machine, the synthesis and back-end scripts take 78 hours (3.25 days), 56 hours (2.3 days), and 104 hours (4.3 days) to run the full tool flow for the core, the tile, and the whole chip with

Processor	Architecture	Performance	FPU	OS	MMU	HW Multithreaded	Multicore/ Manycore/GPU	Prototype Core Count	NoC	HDL	Back-end Scripts	License	Last Update
pAVR [57]	8b AVR	Low	✗	✗	✗	✗	No	-	✗	VHDL	✗	GPL v2	Mar 2009
openMSP430 [36, 55]	16bMSP430	Low	✗	✗	✗	✗	No	-	✗	Verilog	✗	BSD	Jul 2015
CPU86 [34]	16b x86	Low	✗	✓	✗	✗	No	-	✗	VHDL	✗	GPL	Jun 2014
Zet [4]	16b x86	Low	✗	✓	✗	✗	No	-	✗	Verilog	✗	GPL v3	Nov 2013
LatticeMico32 [71]	32b LatticeMico32	Low	✗	✓	✗	✗	No	-	✗	Verilog	✗	GPL	Oct 2015
ZPU [5]	32b MIPS	Low	✗	✓	✗	✗	No	-	✗	VHDL	✗	FreeBSD & GPL	Apr 2015
SecretBlaze [11]	32b MicroBlaze	Low	✗	✗	✗	✗	No	-	✗	VHDL	✗	GPL v3	Dec 2012
AltOr32 [53]	32b ORBIS	Low	✗	✓	✗	✗	No	-	✗	Verilog	✗	LGPL v3	Feb 2015
aeMB [8, 51, 73]	32b MicroBlaze	Medium	✗	✓	✗	✓	No	-	✗	Verilog	✗	LGPL v3	Feb 2012
Amber [54]	32b ARM v2a	Medium	✗	✓	✗	✗	No	-	✗	Verilog	✗	LGPL	Nov 2015
OpenRISC [56, 79]	32b/64b ORBIS	Medium	✓	✓	✓	✗	No	-	✗	Verilog	✗	LGPL	Dec 2012
MIPS32 r1 [3]	32b MIPS32 r1	Medium	✗	✓	✗	✓	No	-	✗	Verilog	✗	LGPL v3	Jul 2015
LEON 3 [24, 65]	32b SPARC V8	Medium	✓($)	✓	✓	✗	SMP/AMP	-	✗	VHDL	✗	GPL	May 2015
OpenScale [18]	32b MicroBlaze	Medium	✗	✓	✗	✗	Manycore	FPGA/6	✓	VHDL	✗	GPL v3	Jan 2012
XUM [2, 47]	32b MIPS32 r2	High	✗	✓	✗	✓	Manycore	FPGA/8	✓	Verilog	✗	LGPL v3	Jul 2015
MIAOW GPGPU [9]	AMD Southern Islands	High	✓	✗	✗	✓	GPU	FPGA/1	✓	Verilog	✗	BSD 3-Clause & GPL v2	Jan 2016
Simply RISC S1 [68]	64b SPARC V9	High	✓	✓	✓	✗	No	-	✗	Verilog	✗	GPL v2	Dec 2008
BERI [1, 83]	64b MIPS/CHERI	High	✓	✓	✓	✓(BERI2)	Multicore	FPGA/4	✗	Bluespec	✗	BERI HW-SW	Jun 2015
OpenSPARC T1/T2 [6, 7]	64b SPARC V9	High	✓	✓	✓	✓	Multicore	ASIC/8	✗	Verilog	✗	GPL v2	2008
RISC-V Rocket [42, 67]	64b scalar RISC-V	High	✓	✓	✓	✗	Manycore	ASIC/2	✓	Chisel	✗	BSD 3-Clause	Jan 2016
RISC-V Boom [22, 66]	64b scalar RISC-V	High	✓	✓	✓	✗	Manycore	FPGA/?	✓	Chisel	✗	BSD 3-Clause	Jan 2016
OpenPiton	64b SPARC V9	High	✓	✓	✓	✓	Manycore	ASIC/25	✓	Verilog	✓	BSD 3-Clause & GPL v2	Jan 2016

Table 4: Taxonomy of differences of open source processors (table data last checked in January 2016).

Submodule	SYN	PAR	STA	ECO	DRC	LVS	Peak
NoC Router	5.38	3.55	0.57	N/A	1.40	1.33	5.38
L1.5	19.91	5.97	1.43	5.00	1.73	1.46	19.91
Core	27.19	7.99	1.96	15.05	2.54	4.03	27.19
Tile	28.04	7.71	1.90	13.62	2.48	8.84	28.04
Chip	8.37	64.54	1.34	64.65	8.54	>41	64.65

Table 6: Peak memory usage (in GByte) of selected synthesis and back-end stages for selected submodules. If any stage executes more than once (STA or for multi-pass flow), the maximum peak usage is shown.

25 tiles respectively. These runtimes are obtained by adding up the run times of all synthesis and back-end stages, from synthesis first pass to DRC and LVS passes. Table 5 details runtimes of individual synthesis and back-end stages for select submodules of the Piton ASIC prototype. For stages in the flow that execute more than once, such as STA or when using a multi-pass flow, the maximum duration for that stage is shown.

Resource requirements for synthesis and back-end scripts, particularly main memory usage, are not excessive. Table 6 (max value is taken for stages that execute more than once) shows that the most memory intensive step of the synthesis and back-end flow requires just over 64GB for the top-level 25-tile chip, making it possible for a high-end desktop at the time of this publication to successfully run through OpenPiton's synthesis and back-end flow.

5. Enabled Applications

Open source platforms such as OpenPiton are designed for flexibility and extensibility, and thus can be easily modified. Researchers leveraging such platforms need not reimplement others' work, which is advantageous for those who lack adequate resources to build infrastructure from scratch. These platforms' independence from a specific target implementation technology also prevents them from going obsolete [80]. In this section, we outline potential applications of the OpenPiton platform in different research domains.

5.1 Architecture

OpenPiton is great tool for computer architecture research and prototyping. In addition to supporting traditional microarchitecture research, OpenPiton offers research opportunities in memory subsystems thanks to its scratch-built uncore. Researchers may also explore manycore scalability, NoCs, and NoC-attached accelerators and I/O.

OpenPiton lowers the barrier to implementing ideas in RTL, while providing accurate estimations of a design's area, power and frequency. The extensive test suite complements the physical implementation by providing behavioral and/or gate-level simulated, and FPGA-emulated validation.

5.2 Operating Systems

While OS researchers have studied how best to add OS support for manycore systems [12, 16, 17, 85], the scarcity of high core-count systems is a barrier to innovation in this space. OpenPiton can enable such research because it can scale to large fabrics while providing support for modern Debian Linux running on an FPGA at tens of MHz.

5.3 Security

Open source processors are widely used by researchers in the security community. While some researchers simulate using open source processors to verify their ideas [13, 77], others build proof-of-concept implementations on FPGA [23, 38, 39, 69, 78]. OpenPiton's substantial test suite, support for an OS and hypervisor, and unique manycore infrastructure, make it a suitable test environment for the security community to further examine novel ideas.

5.4 Compilers

OpenPiton's scalability can assist compiler researchers in understanding how their solutions scale on real hardware. It enables the investigation of future programming models for parallel architectures [62, 64], or how to add automatic parallelization constructs into current languages [19]. The use of the SPARC ISA makes compiler research convenient due to pre-existing compiler support (e.g., GCC).

5.5 Reliability Analysis

Transient and permanent faults can be injected into the RTL of a target design to simulate the effects of faults on system functionality. Consequently, metrics such as architectural vulnerability factor (AVF), failures in time (FIT), and mean time to failure (MTTF) can be estimated. Many studies [25, 26, 48, 60, 61, 89] have used open source processors to perform fault injection. They have also been used in thermal and power management reliability modeling [33, 35] as well as wearout fault modeling studies [74].

With advances in the scalability of reliability analysis tools, researchers can now evaluate the reliability of larger circuits. Moreover, efficient fault simulation schemes have been developed to explore the reliability of multicore and manycore processors [33, 40]. OpenPiton enables researchers to study how different solutions, across the entire design hierarchy, affect the reliability of a manycore. It also facilitates researchers to perform manycore-oriented reliability studies, such as the reliability of NoC architectures [27] and cache coherence protocols.

5.6 Education

Educators have shown the value of using FPGAs in teaching microprocessor architecture [52, 76]. Downloaded from `http://www.openpiton.org`, the OpenPiton platform comes push-button packaged to run on FPGA, including the widely used and subsidized ($160 Academic, currently) Digilent Nexys 4 DDR Artix-7 FPGA Board. In addition, because OpenPiton ships with synthesis and back-end scripts, students learning digital circuits can easily leverage its ASIC flow, in a similar approach to [44], on a real modern manycore processor.

6. Related Work

This section qualitatively compares OpenPiton to other open source platforms available for research implementation and prototyping. Table 4 highlights the main differences between these platforms and OpenPiton.

Most of the open source platforms have rudimentary OS support, frequently only for embedded real time operating systems (RTOS) like eCos [45] by ZPU [5]. Few have memory management units (MMU) needed for full operating system support. In addition, many projects who do support system calls/OSs/memory mapped I/O often require a tethered host core to proxy through. In contrast, leveraging the OpenSPARC T1 core, OpenPiton is a standalone system that has robust OS and hypervisor support.

Most designs in Table 4 are soft-IPs, synthesizable to FPGAs for rapid prototyping, but have not been demonstrated in ASIC. In contrast, OpenPiton supports both FPGA and ASIC usage paradigms. Additional low-end available processors, not shown in Table 4 are mentioned in two surveys [10, 37]. Moreover, technology limitations [88] and the end of Dennard scaling [28] have resulted in a paradigm shift to multi/manycore and heterogeneity [41] and created demand for a truly open source manycore design. Though some designs have limited multicore scalability (i.e. OpenSPARC T1&T2 [6, 7]), OpenPiton is specifically engineered for scalable core count and chip count.

Some recent academic projects [1, 42, 67, 83] are centered around high-level synthesis (HLS) tools, citing the use of past HDL's as archaic and error-prone. In contrast, OpenPiton does not try to innovate in this space. We deliberately use Verilog to leverage industrial-grade design/verification/synthesis flow to significantly reduce design time. Moreover, universal familiarity with Verilog makes OpenPiton easier to access and understand.

While the RISC-V platform provides a crossbar for multicore implementations [67], OpenScale [18], XUM [47], and OpenPiton benefit from scalable mesh networks.

Last, to the best of our knowledge, OpenPiton is the only platform that includes the ASIC back-end (place-and-route) scripts, assisting users with porting the design to other process technologies. Furthermore, the industrial-grade CPU core design together with robust software ecosystem will prove useful to future research projects.

7. Conclusion

The OpenPiton processor design and framework enables researchers to conduct manycore research at-scale and bring novel ideas to implementation using a mature tool chain, a mature ISA, and a complete test suite. OpenPiton can scale from small designs to large designs as proven with implementations in ASIC silicon as well as in many prototyped FPGA boards. OpenPiton is configurable and extensible and comes with a complete verification infrastructure. We are optimistic that OpenPiton will serve as an important tool for scalable manycore processor research and enable more researchers to bring their designs to implementation.

Acknowledgements

This work was partially supported by the NSF under Grants No. CCF-1217553, CCF-1453112, and CCF-1438980, AFOSR under Grant No. FA9550-14-1-0148, and DARPA under Grants No. N66001-14-1-4040 and HR0011-13-2-0005. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

References

[1] Beri processor 'arcina' release 1. `https://github.com/CTSRD-CHERI/beri`. Accessed Jan. 2016.

[2] eXtensible Utah Multicore (xum). `https://github.com/grantae/mips32r1_xum`. Accessed Jan. 2016.

[3] Mips32 release 1. `https://github.com/grantae/mips32r1_core`. Accessed Jan. 2016.

[4] Zet processor. `http://zet.aluzina.org/index.php/Zet_processor`. Accessed Jan. 2016.

[5] Zylin cpu. `https://github.com/zylin/zpu`. Accessed Jan. 2016.

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

[7] *OpenSPARC T2 Core Microarchitecture Specification*. Santa Clara, CA, 2007.

[8] Aeste Works. Aemb multi-threaded 32-bit embedded core family. `https://github.com/aeste/aemb`. Accessed Jan. 2016.

[9] 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.

[10] R. R. Balwaik, S. R. Nayak, and A. Jeyakumar. Open-source 32-bit risc soft-core processors. *IOSR Journal od VLSI and Signal Processing*, 2(4):43–46, 2013.

[11] L. Barthe, L. Cargnini, P. Benoit, and L. Torres. The secretblaze: A configurable and cost-effective open-source soft-core processor. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 310–313, May 2011.

[12] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. The multikernel: A new os architecture for scalable multicore systems. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 29–44, New York, NY, USA, 2009. ACM.

[13] M. Bilzor, T. Huffmire, C. Irvine, and T. Levin. Evaluating security requirements in a general-purpose processor by combining assertion checkers with code coverage. In *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, pages 49–54, June 2012.

[14] 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. *SIGARCH Computer Architecture News*, 39(2):1–7, Aug. 2011.

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

[16] S. Boyd-Wickizer, H. Chen, R. Chen, Y. Mao, F. Kaashoek, R. Morris, A. Pesterev, L. Stein, M. Wu, Y. Dai, Y. Zhang, and Z. Zhang. Corey: An operating system for many cores. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 43–57, Berkeley, CA, USA, 2008. USENIX Association.

[17] S. Boyd-Wickizer, A. T. Clements, Y. Mao, A. Pesterev, M. F. Kaashoek, R. Morris, and N. Zeldovich. An analysis of linux scalability to many cores. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

[18] R. Busseuil, L. Barthe, G. Almeida, L. Ost, F. Bruguier, G. Sassatelli, P. Benoit, M. Robert, and L. Torres. Open-scale: A scalable, open-source noc-based mpsoc for design space exploration. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 Int. Conference on*, pages 357–362, Nov 2011.

[19] S. Campanoni, T. Jones, G. Holloway, V. J. Reddi, G.-Y. Wei, and D. Brooks. Helix: Automatic parallelization of irregular programs for chip multiprocessing. In *Proceedings of the Tenth International Symposium on Code Generation and Optimization*, CGO '12, pages 84–93, New York, NY, USA, 2012. ACM.

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

[21] 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*, SC '11, pages 52:1–52:12, New York, NY, USA, 2011. ACM.

[22] C. Celio, D. A. Patterson, and K. Asanović. The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor. Technical Report UCB/EECS-2015-167, EECS Department, University of California, Berkeley, Jun 2015.

[23] D. Champagne and R. Lee. Scalable architectural support for trusted software. In *High Performance Computer Architecture (HPCA), IEEE 16th Int. Symposium on*, pages 1–12, Jan 2010.

[24] Cobham Gaisler AB. Grlib ip core users manual. May 2015.

[25] A. da Silva and S. Sanchez. Leon3 vip: A virtual platform with fault injection capabilities. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 813–816, Sept 2010.

[26] M. Ebrahimi, L. Chen, H. Asadi, and M. Tahoori. Class: Combined logic and architectural soft error sensitivity analysis. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 601–607, Jan 2013.

[27] M. Ebrahimi, M. Daneshtalab, and J. Plosila. High performance fault-tolerant routing algorithm for noc-based many-core systems. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 462–469, Feb 2013.

[28] 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.

[29] 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.

[30] 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.

[31] Y. Fu and D. Wentzlaff. Prime: A parallel and distributed simulator for thousand-core chips. In *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on*, pages 116–125, March 2014.

[32] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, May 1996.

[33] M.-Y. Hsieh. A scalable simulation framework for evaluating thermal management techniques and the lifetime reliability of multithreaded multicore systems. In *Int. Green Computing Conference and Workshops*, pages 1–6, July 2011.

[34] HT-Lab. Cpu86: 8088 fpga ip core. `http://ht-lab.com/freecores/cpu8086/cpu86.html`. Accessed Jan. 2016.

[35] H. Hua, C. Mineo, K. Schoenfliess, A. Sule, S. Melamed, R. Jenkal, and W. Davis. Exploring compromises among timing, power and temperature in three-dimensional integrated circuits. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 997–1002, 2006.

[36] T. Instruments. Msp430x1xx family users guide, 2006.

[37] R. Jia, C. Lin, Z. Guo, R. Chen, F. Wang, T. Gao, and H. Yang. A survey of open source processors for fpgas. In *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, pages 1–6, Sept 2014.

[38] O. Khalid, C. Rolfes, and A. Ibing. On implementing trusted boot for embedded systems. In *Hardware-Oriented Security and Trust, IEEE Int. Symposium on*, pages 75–80, June 2013.

[39] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou. Designing and implementing malicious hardware. *LEET*, 8:1–8, 2008.

[40] M. Kochte, M. Schaal, H. Wunderlich, and C. Zoellin. Efficient fault simulation on many-core processors. In *ACM/IEEE Design Automation Conference*, pages 380–385, June 2010.

[41] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-isa heterogeneous multi-core architectures: The potential for processor power reduction. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM Int. Symposium on*, pages 81–92. IEEE, 2003.

[42] Y. Lee, A. Waterman, R. Avizienis, H. Cook, C. Sun, V. Stojanovic, and K. Asanović. A 45nm 1.3ghz 16.7 double-precision gflops/w risc-v processor with vector accelerators. In *European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014 - 40th*, pages 199–202, Sept 2014.

[43] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM Int. Symposium on*, pages 469–480, Dec 2009.

[44] J. Lu and B. Taskin. From rtl to gdsii: An asic design course development using synopsys® university program. In *Microelectronic Systems Education (MSE), 2011 IEEE International Conference on*, pages 72–75, June 2011.

[45] A. J. Massa. *Embedded software development with eCos*. Prentice Hall Professional, 2003.

[46] 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.

[47] B. L. Meakin. Multicore system design with xum: The extensible utah multicore project. Master's thesis, The University of Utah, 2010.

[48] N. Mehdizadeh, M. Shokrolah-Shirazi, and S. Miremadi. Analyzing fault effects in the 32-bit openrisc 1200 microprocessor. In *Availability, Reliability and Security. ARES 08. Third International Conference on*, pages 648–652, March 2008.

[49] 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.

[50] J. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal. Graphite: a distributed parallel simulator for multicores. In *IEEE 16th Int. Symp. on High Performance Computer Architecture*, pages 1–12, 2010.

[51] S. T. S. Ngiap. Aemb 32-bit microprocessor core datasheet, November 2007.

[52] J. Olivares, J. Palomares, J. Soto, and J. Gámez. Teaching microprocessors design using fpgas. In *Education Engineering (EDUCON), 2010 IEEE*, pages 1189–1193, April 2010.

[53] OpenCores. Altor32 - alternative lightweight openrisc cpu. `http://opencores.org/project,altor32`. Accessed Jan. 2016.

[54] OpenCores. Amber arm-compatible core. `http://opencores.org/project,amber`. Accessed Jan. 2016.

[55] OpenCores. Openmsp430. `http://opencores.org/project,openmsp430`. Accessed Jan. 2016.

[56] OpenCores. Or1200 openrisc processor. `http://opencores.org/or1k/OR1200_OpenRISC_Processor`. Accessed Jan. 2016.

[57] OpenCores. pAVR. `http://opencores.org/project,pavr`. Accessed Jan. 2016.

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

[59] P. M. Ortego and P. Sack. Sesc: Superescalar simulator. In *17th Euro micro conf. on real time systems*, pages 1–4, 2004.

[60] I. Parulkar, A. Wood, J. C. Hoe, B. Falsafi, S. V. Adve, J. Torrellas, and S. Mitra. Opensparc: An open platform for hardware reliability experimentation. In *Fourth Workshop on Silicon Errors in Logic-System Effects (SELSE)*. Citeseer, 2008.

[61] A. Pellegrini, R. Smolinski, L. Chen, X. Fu, S. Hari, J. Jiang, S. Adve, T. Austin, and V. Bertacco. Crashtest'ing swat: Accurate, gate-level evaluation of symptom-based resiliency solutions. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 1106–1109, March 2012.

[62] C. D. Polychronopoulos. *Parallel programming and compilers*, volume 59. Springer Science & Business Media, 2012.

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

[64] A. Raman, A. Zaks, J. W. Lee, and D. I. August. Parcae: A system for flexible parallel execution. In *Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 133–144, New York, NY, USA, 2012.

[65] Aeroflex Gaisler AB. Sparc v8 32-bit processor leon3/leon3-ft companioncore data sheet, March 2010.

[66] UC Berkeley Architecture Research. The berkeley out-of-order risc-v processor. `https://github.com/ucb-bar/riscv-boom`. Accessed Jan. 2016.

[67] UC Berkeley Architecture Research. Rocket core. `https://github.com/ucb-bar/rocket`. Accessed Jan. 2016.

[68] S. RISC. Simply risc s1 core. `http://www.srisc.com/?s1`. Accessed Jan. 2016.

[69] P. Schaumont and I. Verbauwhede. Thumbpod puts security under your thumb. *Xilinx® Xcell J*, 2003.

[70] 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.

[71] L. Semiconductor. Latticemico32 open, free 32-bit soft processor. `http://www.latticesemi.com/en/Products/DesignSoftwareAndIP/IntellectualProperty/IPCore/IPCores02/LatticeMico32.aspx`. Accessed Jan. 2016.

[72] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks. Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures. In *The 41st Annual Int. Symposium on Computer Architecture*, pages 97–108, Piscataway, NJ, USA, 2014. IEEE Press.

[73] S. Shengfeng, Z. Dexue, and Y. Guoping. Soc verification platform based on aemb softcore processor [j]. *Microcontrollers & Embedded Systems*, 4:016, 2010.

[74] J. C. Smolens, B. T. Gold, J. C. Hoe, B. Falsafi, and K. Mai. Detecting emerging wearout faults. In *Proc. of Workshop on SELSE*, 2007.

[75] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path oram: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, pages 299–310, New York, NY, USA, 2013. ACM.

[76] A. Strelzoff. Teaching computer architecture with fpga soft processors. In *ASEE Southeast Section Conference*, 2007.

[77] J. Szefer and R. Lee. Architectural support for hypervisor-secure virtualization. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, pages 437–450, New York, NY, USA, 2012. ACM.

[78] 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.

[79] J. Tandon. The openrisc processor: open hardware and linux. *Linux Journal*, 2011(212):6, 2011.

[80] J. Tong, I. Anderson, and M. Khalid. Soft-core processors for embedded systems. In *Microelectronics, 2006. ICM '06. International Conference on*, pages 170–173, Dec 2006.

[81] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli. Multi2sim: A simulation framework for cpu-gpu computing. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT '12, pages 335–344, New York, NY, USA, 2012. ACM.

[82] 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.

[83] R. N. M. Watson, J. Woodruff, D. Chisnall, B. Davis, W. Koszek, A. T. Markettos, S. W. Moore, S. J. Murdoch, P. G. Neumann, R. Norton, and M. Roe. Bluespec Extensible RISC Implementation: BERI Hardware reference. Technical Report UCAM-CL-TR-868, University of Cambridge, Computer Laboratory, Apr. 2015.

[84] 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.

[85] D. Wentzlaff, C. Gruenwald, III, N. Beckmann, K. Modzelewski, A. Belay, L. Youseff, J. Miller, and A. Agarwal. An operating system for multicore and clouds: Mechanisms and implementation. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 3–14, New York, NY, USA, 2010. ACM.

[86] 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.

[87] 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.

[88] D. Yeh, L.-S. Peh, S. Borkar, J. Darringer, A. Agarwal, and W.-M. Hwu. Thousand-core chips [roundtable]. *Design Test of Computers, IEEE*, 25(3):272–278, May 2008.

[89] M. Zandrahimi, H. Zarandi, and A. Rohani. An analysis of fault effects and propagations in zpu: The world's smallest 32 bit cpu. In *Quality Electronic Design (ASQED), 2010 2nd Asia Symp. on*, pages 308–313, Aug 2010.