A 45nm 1.3GHz 16.7 Double-Precision GFLOPS/W RISC-V Processor with Vector Accelerators

Yunsup Lee*, Andrew Waterman*, Rimas Avizienis*, Henry Cook*, Chen Sun*†,
Vladimir Stojanović*†, Krste Asanović*
Email: {yunsup, waterman, rimas, hcook, vlada, krste}@eecs.berkeley.edu, sunchen@mit.edu
*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA
†Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA

***Abstract*—A 64-bit dual-core RISC-V processor with vector accelerators has been fabricated in a 45 nm SOI process. This is the first dual-core processor to implement the open-source RISC-V ISA designed at the University of California, Berkeley. In a standard 40 nm process, the RISC-V scalar core scores 10% higher in DMIPS/MHz than the Cortex-A5, ARM's comparable single-issue in-order scalar core, and is 49% more area-efficient. To demonstrate the extensibility of the RISC-V ISA, we integrate a custom vector accelerator alongside each single-issue in-order scalar core. The vector accelerator is 1.8× more energy-efficient than the IBM Blue Gene/Q processor, and 2.6× more than the IBM Cell processor, both fabricated in the same process. The dual-core RISC-V processor achieves maximum clock frequency of 1.3 GHz at 1.2 V and peak energy efficiency of 16.7 double-precision GFLOPS/W at 0.65 V with an area of 3 mm^2.**

I. Introduction

As we approach the end of conventional transistor scaling, computer architects are forced to incorporate specialized and heterogeneous accelerators into general-purpose processors for greater energy efficiency. Many proposed accelerators, such as those based on GPU architectures, require a drastic reworking of application software to make use of separate ISAs operating in memory spaces disjoint from the demand-paged virtual memory of the host CPU. RISC-V [1] is a new completely open general-purpose instruction set architecture (ISA) developed at the University of California, Berkeley, which is designed to be flexible and extensible to better integrate new efficient accelerators close to the host cores. The open-source RISC-V software toolchain includes a GCC cross-compiler, an LLVM cross-compiler, a software ISA simulator, an ISA verification suite, a Linux port, and additional documentation, and is available at `www.riscv.org`.

In this paper, we present a 64-bit dual-core RISC-V processor with custom vector accelerators in a 45 nm SOI process. Our RISC-V scalar core achieves 1.72 DMIPS/MHz, outperforming ARM's Cortex-A5 score of 1.57 DMIPS/MHz by 10% in a smaller footprint. Our custom vector accelerator is 1.8× more energy-efficient than the IBM Blue Gene/Q processor and 2.6× more than the IBM Cell processor for double-precision floating-point operations, demonstrating that high efficiency can be obtained without sacrificing a unified demand-paged virtual memory environment.

II. Chip Architecture

Figure 1 shows the block diagram of the dual-core processor. Each core incorporates a 64-bit single-issue in-order Rocket scalar core, a 64-bit Hwacha vector accelerator, and their associated instruction and data caches, as described below.

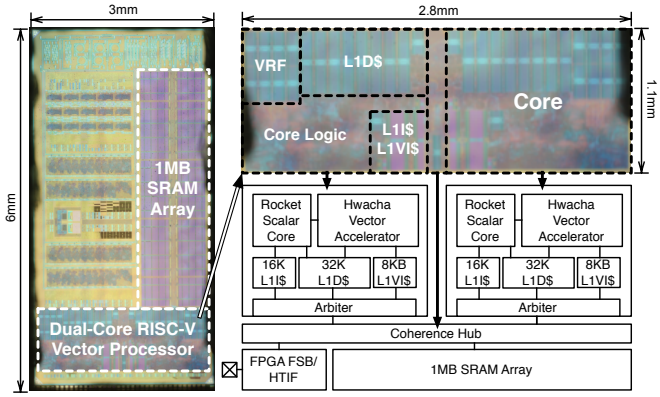

Fig. 1. Backside chip micrograph (taken with a removed silicon handle) and processor block diagram.

A. Rocket Scalar Core

Rocket is a 6-stage single-issue in-order pipeline that executes the 64-bit scalar RISC-V ISA (see Figure 2). The scalar datapath is fully bypassed but carefully designed to minimize the impact of long clock-to-output delays of compiler-generated SRAMs in the caches. A 64-entry branch target buffer, 256-entry two-level branch predictor, and return address stack together mitigate the performance impact of control hazards. Rocket implements an MMU that supports page-based virtual memory and is able to boot modern operating systems, including Linux. Both caches are virtually indexed physically tagged with parallel TLB lookups. The data cache is non-blocking, allowing the core to exploit memory-level parallelism.

Rocket has an optional IEEE 754-2008-compliant FPU, which can execute single- and double-precision floating-point operations, including fused multiply-add (FMA), with hardware support for denormals and other exceptional values. The

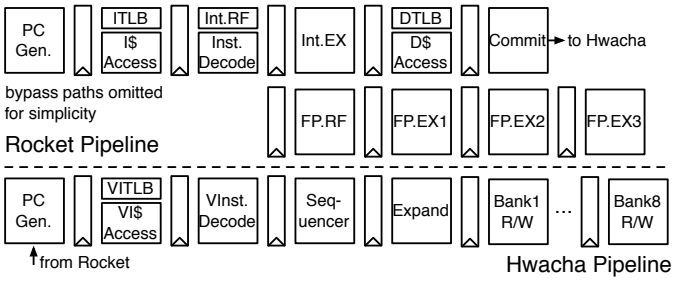

Fig. 2. Rocket scalar plus Hwacha vector pipeline diagram.

978-1-4799-5696-8/14/$31.00 ©2014 IEEE

TABLE I. ARM CORTEX-A5 VS. RISC-V ROCKET

	ARM Cortex-A5 [2]	RISC-V Rocket
Process	TSMC40GPLUS	
Dhrystone Performance	1.57 DMIPS/MHz	1.72 DMIPS/MHz
ISA Register Width	32 bits	64 bits
Frequency	>1 GHz	>1 GHz
Area excluding caches	0.27 mm^2	0.14 mm^2
Area with 16 KB caches	0.53 mm^2	0.39 mm^2
Area Efficiency	2.96 DMIPS/MHz/mm^2	4.41 DMIPS/MHz/mm^2
Dynamic Power	<0.08 mW/MHz	0.034 mW/MHz

fully-pipelined double-precision FMA unit has a latency of three clock cycles.

To compare against published numbers for a 32-bit ARM Cortex-A5, we implemented a similarly configured 64-bit Rocket core and evaluated it with the same Dhrystone benchmark and the same TSMC40GPLUS process corner that ARM used to evaluate the Cortex-A5 [2]. As shown in Table I, Rocket attains greater performance (1.72 vs. 1.57 DMIPS/MHz) in substantially less area (0.39 vs. 0.53 mm^2 including caches) and dynamic power (0.034 vs. 0.08 mW/MHz).

B. Hwacha Vector Accelerator

Hwacha is a single-lane decoupled vector pipeline optimized for an ASIC process. Hwacha more closely resembles traditional Cray vector pipelines [3] than the SIMD units in SSE, AVX, or NEON. For greater efficiency than traditional vector architectures, vector arithmetic instructions are packed into separate blocks of instructions that are queued to be fetched and decoded only after vector memory instructions have run ahead to prefetch the needed data [4].

Figure 3 shows the microarchitecture of the Hwacha vector accelerator. Rocket sends Hwacha a vector instruction at the scalar commit stage once all exceptions are cleared. The Hwacha issue unit fetches and decodes a vector instruction, and waits until all hazards (e.g., data hazards, structural hazards, bank conflicts) are cleared. Once all hazards are cleared, the vector instruction is issued to the sequencer. The sequencer holds a decoded vector instruction until all elements in a vector are executed. The expander breaks vector instructions

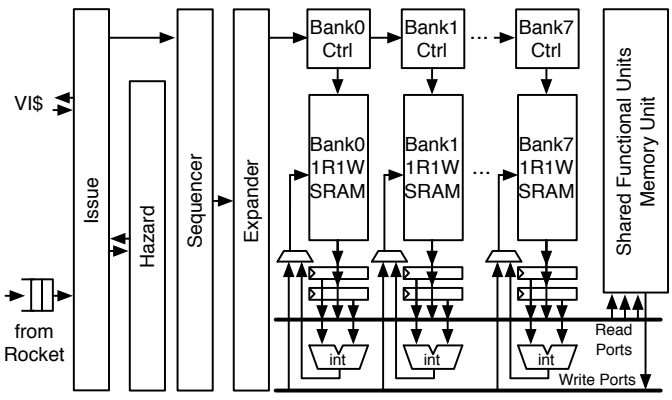

Fig. 3. Hwacha Vector Accelerator Microarchitecture. Replicated structures of bank 2 to 6 are omitted in the diagram.

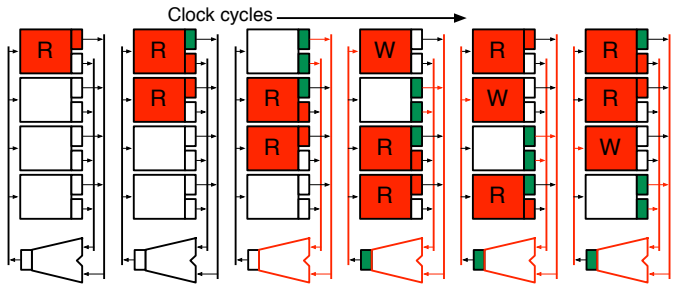

Fig. 4. Banked VRF execution example showing how a banked VRF is effectively able to read out 2 operands/cycle from each bank after a 2-cycle initial startup latency. Each square represents a 1R1W SRAM macro (same 1R1W SRAM also shown in Figure 3), and small rectangles represent operand registers (two registers below the 1R1W SRAM in Figure 3). For simplicity, the example depicts only 4 banks.

into bank micro-ops, which are the unit of execution on the lane. Each bank micro-op executes on each bank and is then passed systolically to the next bank as shown in Figure 3. To improve area efficiency, we split the vector register file (VRF) into 8 banks of 1R1W SRAM macros, avoiding expensive flop-based multiport register files. By exploiting the regular operand access pattern (Figure 4), we reduce the impact of banking on performance. Per-bank integer ALUs are directly connected to the read and write ports of each bank to eliminate structural hazards on integer execution units, to reduce operand energy, and to increase integer performance. Larger functional units, such as the 64-bit integer multiplier and single- and double-precision floating-point units, are shared across all banks. To further reduce area, the floating-point functional units are shared with Rocket [4]. The memory unit supports unit-strided and strided vector memory accesses, as well as vector gathers and scatters. Virtual memory with restartable exceptions [5] is also supported in the Hwacha vector accelerator.

C. Memory System

Each Rocket core is attached to a 2-way set-associative 16 KB instruction cache, and each Hwacha vector accelerator is likewise attached to its own direct-mapped 8 KB instruction cache. A 4-way set-associative 32 KB non-blocking data cache is shared between the scalar core and the vector accelerator. These private L1 data caches are kept coherent via a broadcast-based MESI cache coherence protocol. The protocol also keeps the caches coherent with respect to memory requests from the host-target interface block (HTIF). The coherence protocol is implemented by a hub controller, which tracks outstanding coherence transactions, and a set of crossbar networks, one for each coherence protocol message type.

In high-performance testing mode, the 1 MB SRAM array, which emulates DRAM timing, is used as the only backing memory. In basic testing mode, main memory traffic traverses the FPGA FSB, which uses a low-bandwidth source-synchronous protocol to communicate with an external FPGA that supports up to 4 GB of backing memory over the chip I/O.

D. System Interface

The host-target interface block (HTIF) uses the same protocol and physically shares the same chip I/O as the FPGA

FSB to allow the host to read and write target memory and control registers. We use this interface to download and run programs and to service OS system calls such as file I/O.

III. Chip Implementation

A. RTL Development and Verification

The processor RTL is written in Chisel [6], a new hardware construction language that lets designers express hardware as highly parameterized generators to aid design tuning under different area, performance, power, and process constraints. Processor parameters such as the number of cores, cache sizes, associativity, number of TLB entries, number of floating-point pipeline stages, cache-coherence protocol, and width of off-chip I/O are selected during elaboration via a configuration object. Chisel generates both a C++ cycle-accurate software simulator and low-level synthesizable Verilog that maps to standard FPGA or ASIC flows. We develop and verify the RTL using the C++ cycle simulator through both directed and randomly generated test programs. The RTL is also mapped to an FPGA for validation on longer programs, such as the Linux kernel.

B. Physical Design Flow

The Synopsys ASIC CAD toolflow (Design Compiler, IC Compiler) is used to map the Chisel-generated Verilog to a multi-Vt standard cell library and memory-compiler-generated SRAMs in a 45 nm SOI technology. Hierarchical mapping is used: the core processor module is synthesized and placed-and-routed independently, and two instances of this module are instantiated at the top level of the design. This bottom-up approach significantly reduces overall CAD tool runtime and should scale well to designs that incorporate larger numbers of cores. The final signoff steps are also performed using Synopsys tools, and include static timing analysis (PrimeTime) with extracted RC (StarRC), formal verification (Formality) comparing the gate-level netlist against the RTL, post place-and-route gate-level simulation (VCS) with all nets back-annotated with delay calculated with extracted RC. The flow is highly automated, which enables us to quickly iterate through physical design variations and arrive at an acceptable QoR. The chip utilizes C4 area I/O and is flip-chip bonded to a PCB for testing.

IV. Chip Results and Measurements

Figure 5 shows the measurement setup. The processor talks to a Virtex 6 FPGA over a custom front-side bus at a maximum

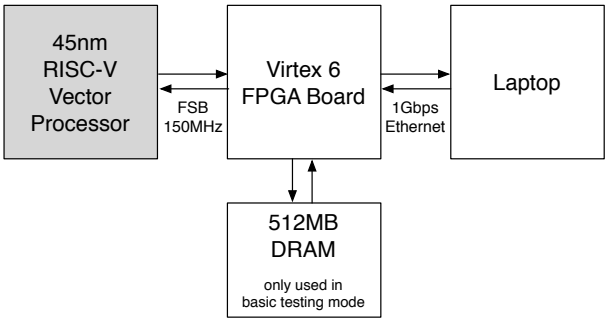

Fig. 5. Measurement setup.

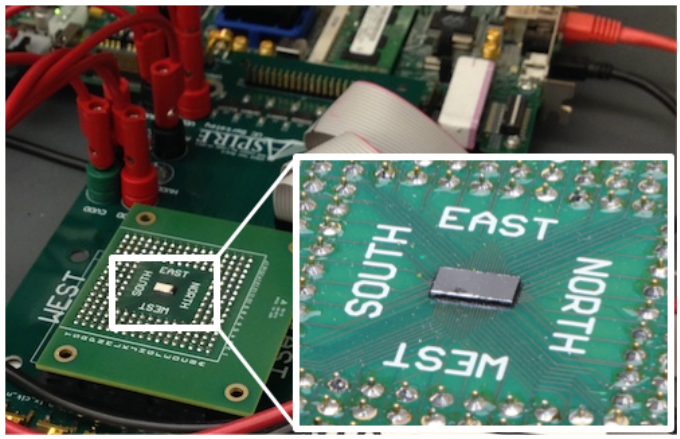

Fig. 6. Photo of measurement setup. The processor is flip-chip mounted on the board as shown in the inset.

TABLE II. Chip Parameters

Process	45 nm SOI CMOS, 11 metal layers	
Size	Processor	2.8 mm (W) × 1.1 mm (H)
	1 Core	1.37 mm (W) × 1.06 mm (H)
	SRAM Array	1.1 mm (W) × 4 mm (H)
Std. Cells	Processor	425K (85K flip-flops)
	1 Core	192K (36K flip-flops)
SRAM Bits	Processor	1246K
	1 Core	621K
Frequency	1 GHz (Nominal), 250 MHz-1.3 GHz	
Voltage	1 V (Nominal), 0.65 V-1.2 V	
Power	300 mW-430 mW (Nominal), 40 mW-960 mW	

speed of 150 MHz. The 512 MB DRAM connected to the FPGA serves as main memory when the processor is used in basic testing mode. The on-chip 1 MB SRAM array acts as main memory in high-performance testing mode. A laptop can read and write target memory and per-core control registers via an Ethernet link connected to the FPGA, and is used to download and run programs. A photo of the measurement setup is shown in Figure 6. Table II summarizes the resulting chip parameters.

A. Area, Frequency, and Power

The dual-core RISC-V processor's area is approximately 3 mm^2. Each core occupies about 1.45 mm^2, and the remainder is occupied by the uncore logic. At the nominal supply voltage of 1 V, the processor operates at a nominal frequency of 1 GHz. The voltage can be scaled down to 0.65 V, at which point the chip runs at 250 MHz. The processor runs at a maximum clock frequency of 1.3 GHz at 1.2 V, at which point the SRAM array becomes the speed-limiting factor. The power consumption varies from 300 mW to 430 mW at 1 V depending on the processor activity, and scales from 40 mW at 0.65 V up to 960 mW at 1.2 V.

B. Energy Efficiency

To measure the resulting energy efficiency, we use a double-precision floating-point matrix-multiplication kernel that runs on Hwacha at 78% of peak. Using the high-performance testing mode, we measure power consumption

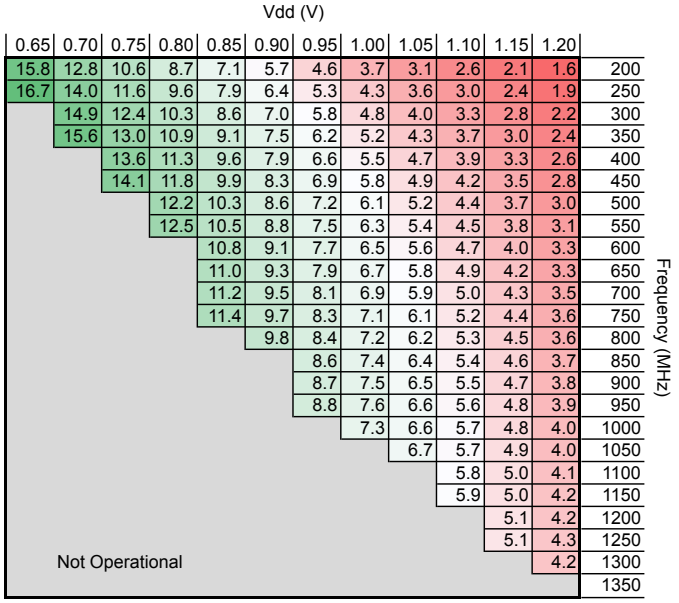

Fig. 7. Shmoo plot of double-precision GFLOPS/W.

from the processor power supply, which includes the IR drop due to the non-ideal power delivery. The resulting shmoo plot of double-precision GFLOPS/W running the matrix-multiplication kernel is shown in Figure 7. Through supply scaling, we achieve a peak energy efficiency of 16.7 double-precision GFLOPS/W at 0.65 V and 250 MHz. At 0.8 V and 1 V, we achieve 12.5 GFLOPS/W at 550 MHz and 7.3 GFLOPS/W at 1 GHz respectively.

Detailed power breakdowns of commercial processors are rarely available, and so comparing processor power consumption is challenging. With that caveat, Table III compares our energy efficiency with the Blue Gene/Q and IBM Cell processors, both fabricated in the same 45 nm SOI process. Blue Gene/Q can compute 204.8 GFLOPS in 55 W at 0.8 V [7][8]. Its cores dissipate 54% of the total power [9]. Conservatively assuming that a double-precision floating-point matrix-multiplication kernel can achieve peak floating-point performance of 204.8 GFLOPS, Blue Gene/Q achieves 6.9 GFLOPS/W—55% of our energy efficiency at 0.8 V. The Cell processor [10] optimized for double-precision achieves 108.8 GFLOPS while dissipating 75 W in 65 nm SOI [11]. The single-precision optimized Cell processor reported a 40% power reduction from 65 nm to 45 nm [12]. Conservatively assuming peak floating-point performance and half of the power going to the cores, the Cell processor would achieve 4.8 GFLOPS/W at 0.8 V in 45 nm SOI—38% of our energy efficiency at the same voltage.

TABLE III. ENERGY EFFICIENCY COMPARISON AT 0.8 V IN 45NM SOI

	Frequency (GHz)	64-bit GFLOPS	Power (W)	Efficiency (GFLOPS/W)
Blue Gene/Q [7][8][9]	1.60	204.8	29.7	6.9
Cell [10][11][12]	3.20	108.8	22.5	4.8
This Work	0.55	1.72	0.138	12.5

V. CONCLUSION

The dual-core RISC-V processor demonstrates that vector accelerators can be integrated in a demand-paged virtual memory environment with high energy efficiency. This dual-core chip achieves a peak energy efficiency of 16.7 double-precision GFLOPS/W with a processor area of 3 mm^2 in a 45 nm SOI process, and runs at a maximum clock frequency of 1.3 GHz. We also show that the open-source RISC-V ISA can serve as a competitive base ISA for integrating specialized heterogeneous accelerators. We plan to open-source our RISC-V processor implementations at `www.riscv.org`.

ACKNOWLEDGMENT

This work was funded by DARPA award HR0011-11-C-0100 under the POEM program led by Dr. Jagdeep Shah, DARPA award HR0011-12-2-0016, the Center for Future Architecture Research, a member of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, an NVIDIA graduate fellowship, and ASPIRE Lab industrial sponsors and affiliates Intel, Google, Nokia, NVIDIA, Oracle, and Samsung. Any opinions, findings, conclusions, or recommendations in this paper are solely those of the authors and does not necessarily reflect the position or the policy of the sponsors. We gratefully acknowledge the support of all POEM team members at MIT, UC Berkeley, and CU Boulder.

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović, "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54, May 2014. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html

[2] "ARM Cortex-A5 Processor Performance and Specification," http://www.arm.com/products/processors/cortex-a/cortex-a5.php.

[3] R. M. Russell, "The Cray-1 Computer System," *Communications of the ACM*, vol. 21, no. 1, pp. 63–72, Jan 1978.

[4] Y. Lee, R. Avizienis, A. Bishara, R. Xia, D. Lockhart, C. Batten, and K. Asanović, "Exploring the Tradeoffs Between Programmability and Efficiency in Data-Parallel Accelerators," *ACM Trans. Comput. Syst.*, vol. 31, no. 3, pp. 6:1–6:38, Aug 2013.

[5] H. Vo, Y. Lee, A. Waterman, and K. Asanović, "A Case for OS-Friendly Hardware Accelerators," *Workshop on the Interaction between Operating System and Computer Architecture (WIVOSCA), at the International Symposium on Computer Architecture (ISCA)*, Jun 2013.

[6] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, "Chisel: Constructing Hardware in a Scala Embedded Language," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 1216–1225.

[7] R. Haring, "The Blue Gene/Q Compute Chip," *HotChips 23*, Aug 2011.

[8] IBM Blue Gene team, "Design of the IBM Blue Gene/Q Compute chip," *IBM Journal of Research and Development*, vol. 57, no. 1/2, pp. 1:1–1:13, Jan 2013.

[9] K. Sugavanam, C.-Y. Cher, J. A. Gunnels, R. A. Haring, P. Heidelberger, H. M. Jacobson, M. K. McManus, D. P. Paulsen, D. L. Satterfield, Y. Sugawara, and R. Walkup, "Design for low power and power management in IBM Blue Gene/Q," *IBM Journal of Research and Development*, vol. 57, no. 1/2, pp. 3:1–3:11, Jan 2013.

[10] J. Kahle, M. Day, H. Hofstee, C. Johns, T. R. Maeurer, and D. Shippy, "Introduction to the Cell multiprocessor," *IBM Journal of Research and Development*, vol. 49, no. 4.5, pp. 589–604, July 2005.

[11] D. Grice, "Cell/B.E. Processor-Based Systems and Software Offerings," *Annual Meeting of ScicomP*, 2008.

[12] O. Takahashi *et al.*, "Migration of Cell Broadband Engine from 65nm SOI to 45nm SOI," *International Solid-State Circuits Conference*, Feb 2008.