A 16mm^2 106.1 GOPS/W Heterogeneous RISC-V Multi-Core Multi-Accelerator SoC in Low-Power 22nm FinFET

Abraham Gonzalez[1], Jerry Zhao[1], Ben Korpan[1], Hasan Genc[1], Colin Schmidt[1], John Wright[1], Ayan Biswas[1], Alon Amid[1], Farhana Sheikh[2], Anton Sorokin[2], Sirisha Kale[2], Mani Yalamanchi[2], Ramya Yarlagadda[2], Mark Flannigan[3], Larry Abramowitz[2], Elad Alon[1], Yakun Sophia Shao[1], Krste Asanović[1], and Borivoje Nikolić[1]

[1]Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA
[2]Intel Corporation, Hillsboro, OR, USA
[3]Intel Corporation, San Jose, CA, USA

***Abstract*—This work presents a 16mm^2 heterogeneous RISC-V system-on-a-chip (SoC) composed of a high-performance out-of-order core, energy-efficient in-order core, data-parallel vector accelerator, and systolic array deep neural network (DNN) accelerator in low-power Intel 22FFL for general-purpose compute, DNN, and vector workloads. The heterogeneous RISC-V SoC is composed of fully open-source components, including a second-generation Berkeley Out-of-Order Machine (BOOM) with a non-speculative mode attached to a Hwacha vector accelerator, a Rocket in-order core attached to a Gemmini systolic array DNN accelerator, as well as a 1MiB L2 cache and off-chip I/Os. Combined, the variety of heterogeneous compute allows for wide programmability while providing up to a 286x MOPS/W improvement or 282x MOPS improvement over the RISC-V in-order core.**

***Index Terms*—RISC-V, Heterogeneous, Multi-Core, Out-of-Order, Vector, Systolic Array, DNN, Open-Source**

I. Introduction

With the sunset of Dennard scaling, specialized and heterogeneous SoC architectures are the primary method for obtaining high-performance yet energy-efficient SoCs for a diversity of emerging workloads. In particular, compositions of large high-performance processors, small energy-efficient processors, data-parallel ISA extensions, and specialized accelerators have become commonplace. However, prior open-source RISC-V SoCs have not demonstrated such heterogeneous architectures, with associated test chips primarily implementing homogeneous architectures [1]–[3]. In this work, we demonstrate an open-source, heterogeneous RISC-V SoC that combines an out-of-order core, an in-order core, a vector accelerator, and a systolic array DNN accelerator. By providing flexibility, the SoC targets diverse general-purpose compute, data-parallel workloads, and DNN applications with varying power, performance, and security requirements.

The remainder of this paper is organized as follows. In Section II, we detail the overall system architecture. In Section III, we discuss the design and testing methodology used. Finally, in Section IV, we evaluate the SoC results and conclude in Section V.

II. System Description

The heterogeneous RISC-V SoC includes three main clock and voltage domains as shown in Figure 1 and Figure 2: a high-performance general-application domain with an out-of-order core connected to a vector accelerator, an energy-efficient machine learning (ML) domain with an in-order core connected to a systolic-array-based DNN accelerator, and an uncore domain with a shared L2 cache and peripherals.

Overall	
Process	Intel 22FFL
Max Frequency	961MHz
Clock Domains	3
Voltage Domains	4
Supply Voltage	0.85-1.8V
Die Area	16mm^2
Domain Area	
Gen. App.	2.75mm^2
ML	1.81mm^2
Uncore	2.25mm^2
SRAM Area	
Gen. App.	0.26mm^2
ML	0.75mm^2
Uncore	0.98mm^2

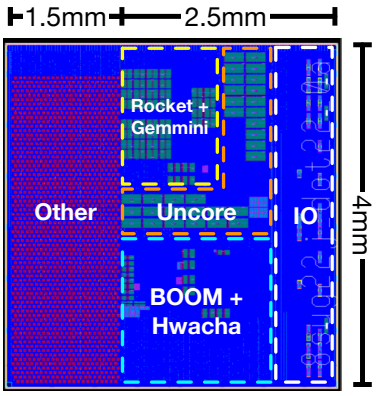

Fig. 1: SoC details and annotated GDS plot.

A. General-Application Domain

The general-application domain, targeting general-purpose and data-parallel vector workloads, includes a BOOM out-of-order processor connected to a Hwacha vector accelerator as shown in Figure 3 [4], [5]. The Berkeley Out-of-Order Core (BOOM) is an open-source superscalar RV64GC RISC-V Linux-capable core generator. The BOOMv2.2 processor, an evolved version of the BOOMv2 architecture, adds dynamically configurable performance gating to prohibit speculation [3]. This allows the core to stall every instruction in issue to ensure that there are no speculation leaks. This is used within Linux to isolate important tasks from certain side-channel attacks and also serves as a design for test (DFT) feature to enable failure isolation. Other new BOOMv2.2 features include support for the compressed (C) RISC-V extension to enable Fedora Linux boot, exposing the Rocket Custom Co-Processor (RoCC) interface to allow connections to custom accelerators, branch prediction performance improvements, map table and freelist selection critical path enhancements, and congestion reduction by reducing signal propagation. The BOOM instance is configured as a 3-wide, 12-stage pipeline that can issue 4 µops per cycle (1 memory, 2 integer, and 1 floating-point). It contains a 32KiB data and instruction cache with a two-level branch predictor consisting of a 8K entry GShare branch predictor with a 2K entry branch target buffer.

Connected over the newly exposed RoCC interface to the BOOM core is an instance of Hwacha, an open-source decoupled vector-fetch accelerator generator. Hwacha is a

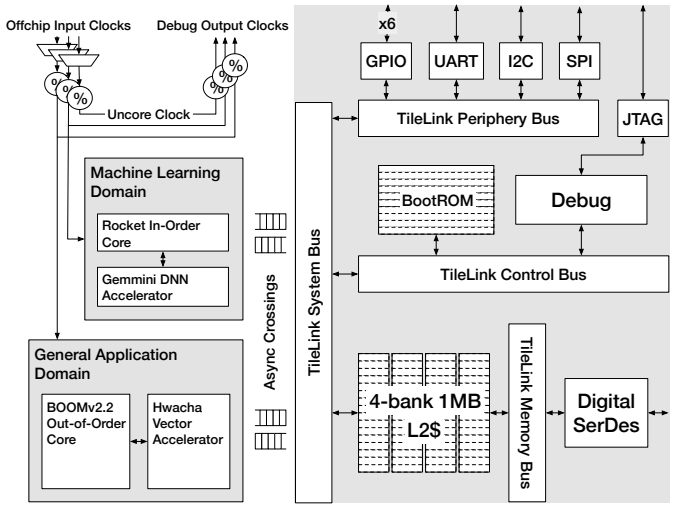

Fig. 2: SoC architecture block diagram. Each grey area indicates a separate clock domain.

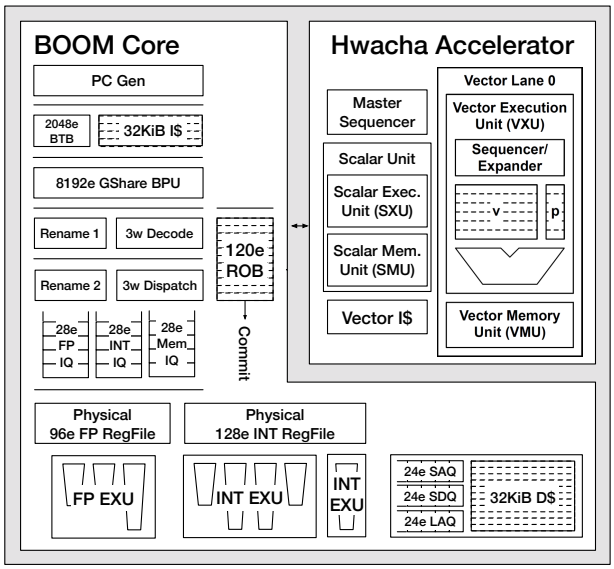

Fig. 3: General-application domain block diagram.

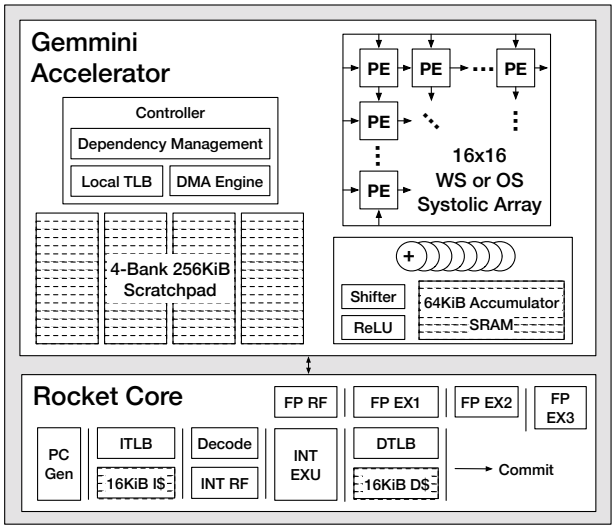

Fig. 4: Machine learning domain block diagram.

vector accelerator generator that supports runtime-configurable vector registers, conditional predication, and virtual memory support. This instance of Hwacha is configured to execute vector operations on double-, single-, and half-precision IEEE floating point, and integers from bytes to double words. It contains 4KiB vector instruction cache with 32 address, 64 scalar, 256 vector, and 16 predication registers. Vector instructions are handled by a single-lane design that splits the instruction into a series of vector µops executed by 4 banks. Lastly, this instance of Hwacha has been extended with new configurable two-dimensional memory and compute operations. These new two-dimensional operations improve energy efficiency on matrix operations particularly for small or non-rectangular matrices.

B. Machine Learning Domain

The ML domain consists of a Rocket in-order processor connected to a Gemmini systolic array DNN accelerator as shown in Figure 4 [6], [7]. The Rocket instance is configured as a small Linux-capable 5-stage in-order RV64GC core. This open-source core includes a 16KiB data and instruction cache as well as a small 2-level branch predictor with a 512 entry branch history table and 14 entry branch target buffer.

Tightly coupled to the Rocket core over its RoCC interface is the open-source Gemmini systolic array DNN accelerator. This Gemmini instance is configured to contain a 16x16 systolic array of 8-bit multiply-accumulate processing elements supporting runtime-programmable weight stationary (WS) and output stationary (OS) dataflows, as well as DNN functional units such as ReLU, ReLU6, and accumulation quantization. Operand matrices are stored in 256KiB of scratchpad memory and 64KiB of accumulator memory.

C. Uncore Domain

Both the general-application and ML domains are connected through asynchronous clock domain crossings to the uncore domain creating 3 separate clock domains. System clocks are provided by a set of clock dividers and multiplexers connected to off-chip single-pin and differential clock inputs. Memory-mapped control registers configure the system clocks and circuitry at runtime. Backing both compute domains is a shared 4-bank 1MiB SiFive open-source L2 cache, which is connected to a low-speed 4-bit-wide digital SerDes port. This low-speed SerDes port is used to connect to backing memory, load small programs into memory, start and end tests, and gain introspection into the test chip. Peripherals of the system include six GPIO, one I2C, one UART, one SPI, and one JTAG port. In addition, off-chip I/Os include clock outputs for each domain and the SerDes clock for visibility. Other off-chip I/Os include a global asynchronous reset signal, multiple clock select signals, and a self-boot signal that indicates whether the chip will execute tests in a tethered mode or not.

III. Design Methodology and Testing

This SoC is the first test chip developed by an early version of the open-source Chipyard SoC development framework [8]. Chipyard provided all the open-source RTL cores, accelerators, caches, and peripherals, and all project-specific improvements were merged into the upstream repository. Together with the SoC integration infrastructure, Chipyard enabled rapid

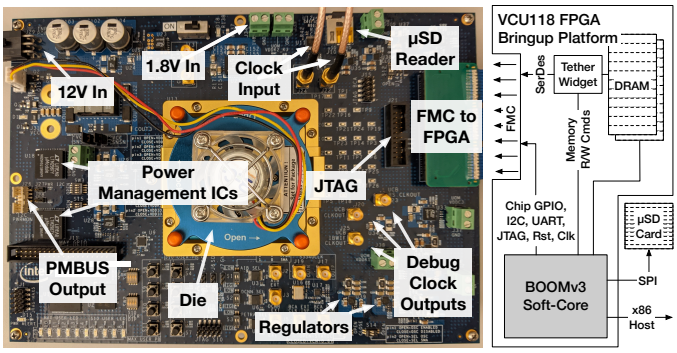

Fig. 5: Annotated board diagram.

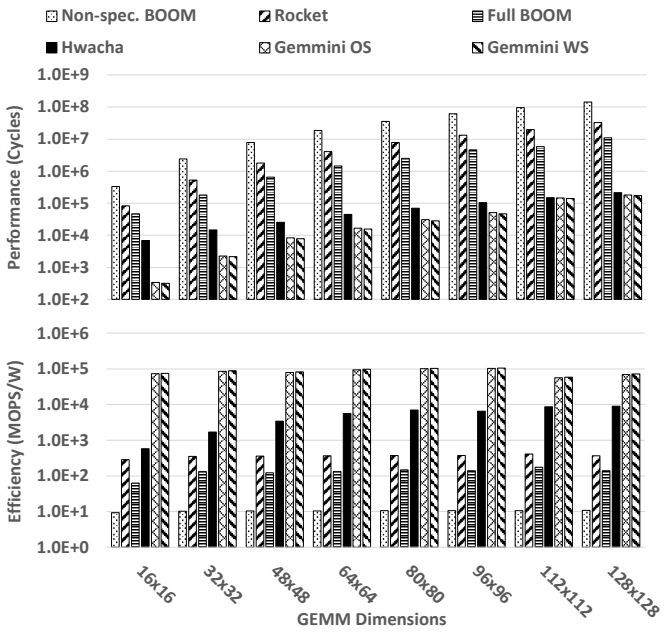

Fig. 6: Performance/efficiency of GEMM benchmarks on each component and mode running at 210MHz.

construction of the heterogeneous SoC. Additionally, Chipyard enabled quick modifications to the SoC module hierarchy by providing design transformation tools such as FIRRTL [9]. This facilitated hardware design modifications to optimize for physical design without the need to change generator source RTL. Chipyard also included various pre-silicon simulation testing flows ranging from small assembly-level tests in RTL-level simulation to large-scale Fedora Linux-level testing with FireSim integrated FPGA-accelerated simulation [10]. In a collaborative process, the SoC design and verification were done by UC Berkeley while the main physical design, packaging, and board design were completed by Intel.

Figure 5 shows the test chip board and FPGA bringup platform that it is connected to. The test chip board provides access to a SPI μSDCard connector used for self-boot, a differential SMA clock input, multiple single-pin SMA debug clock outputs, bypass power connectors, and measurement circuitry for each power domain. Additionally, the main set of I/O signals from the test chip including GPIOs, UART, I2C, low-speed digital SerDes, JTAG, boot/clock select, reset, and single pin low-speed clocks signals are routed to a VITA 57.1 FMC compliant port that connects to a host Xilinx VCU118 FPGA board. This VCU118 FPGA includes a modified Chipyard-based BOOMv3 softcore SoC prototype that interacts with the test chip through the FMC I/Os. By running a modified version of Linux with a custom tether widget, the FPGA system, can provide access to DRAM, probe test chip memory and control registers, and launch tethered programs over the test chip's slow SerDes link. Combined with other Chipyard collateral, the use of this FPGA system allows for quick bringup modifications and easy signal visibility for FMC signals while maintaining compatibility for prior test chips.

IV. Measured Results

Figure 6 demonstrates the heterogeneous nature of the SoC and the ability to run an application across all cores and accelerators with varying levels of efficiency. Specifically, an example general matrix multiplication (GEMM) benchmark, a common kernel of DNN and linear algebra workloads, of increasing size can be executed across the following compute: the BOOM out-of-order core with speculation enabled (full BOOM) and performance gated (non-speculative BOOM), the Rocket in-order core, the Hwacha vector accelerator, and the Gemmini DNN accelerator using a weight stationary (WS) and output stationary (OS) dataflow. When compared to the baseline non-speculative BOOM core performance, the Rocket core obtains a 4.0-4.9x speedup while the full BOOM core obtains 7.1-16.6x speedup. As expected, the data-parallel Hwacha and Gemmini accelerators obtain orders of magnitude greater speedups. Hwacha, coupled to the BOOM core, achieves up to 677x while Gemmini, coupled to the Rocket core, leads in performance with an improvement of 1297.2x by explicitly accelerating GEMMs. The performance gains are echoed in efficiency with the Gemmini WS and OS modes reaching up to 106.1 GOPS/W. Following are Hwacha and Rocket which reach up to 9.0 GOPS/W and 410.2 MOPS/W, respectively. Both the non-speculative BOOM and full BOOM modes are the least efficient at 10.8 MOPS/W and 176.6 MOPS/W, respectively. The low non-speculative BOOM performance is due to stalling instructions in the issue stage which under-utilizes subsequent out-of-order structures. Lower performance of the full BOOM mode is due to systematic front-tend and branch prediction deficiencies that bottleneck IPC and were resolved in a subsequent BOOM version. Finally, measurements were obtained with all cores and accelerators running at 210 MHz at 0.85 V to obtain relative results.

Figure 7 shows the peak frequency and efficiency of a 16x16 GEMM benchmark across the operating voltages of the ML domain. Peak efficiency for Rocket and the Gemmini WS mode occurs at 0.75 V at 385.9 MOPS/W and 73.3 GOPS/W, respectively. Additionally, peak frequencies roughly match across the domain with a range of 675-961 MHz. While the uncore and ML domains were signed off at 500 MHz and the general-application domain was signed-off at 700MHz, the general-application domain contained an unconstrained asynchronous reset path preventing peak frequencies past 210 MHz at any voltage.

Table II demonstrates the performance of both operating modes of BOOM compared to the Rocket core on the CoreMark and Dhrystone benchmarks that fit within the cores'

	This Work	ISSCC'21 [1]	VLSI'19 [11]	TVLSI'2019 [2]	VLSI'18 [3]
Process	22nm FinFET	16nm FinFET	16nm FinFET	22nm FD-SOI	28nm CMOS
Die Size	16mm^2	24.01mm^2	25mm^2	9mm^2	4.86mm^2
Base ISA	RV64GC	RV64G	ARMv8-A	RV64IMC	RV64G
Energy Efficiency	106.1 GOPS/W	209.5 GH-FLOPS/W	58.7 GOPS/W	40 GOPS/W	-
Max Frequency	961MHz (ML Domain) 210MHz (Gen. App. Domain)	1.44GHz	>1GHz	1.7GHz	1GHz
Vector Support	Hwacha4p5	Hwacha4	NEON	-	-
DNN Acceleration	Gemmini	-	CCA	-	-
Memory System	1MiB L2	4 256KiB L2s, 3MiB L3	4MiB Scratchpad	520KiB Scratchpad	1MiB L2
CoreMark/MHz	2.37	-	-	-	3.77
DMIPS/MHz	2.13	-	-	1.65	-
Open-Source RTL	Yes	Yes	No	Yes	Partially
Processor Type	In-Order/Out-of-Order	In-Order	In-Order	In-Order	Out-of-Order

TABLE I: Comparison Table.

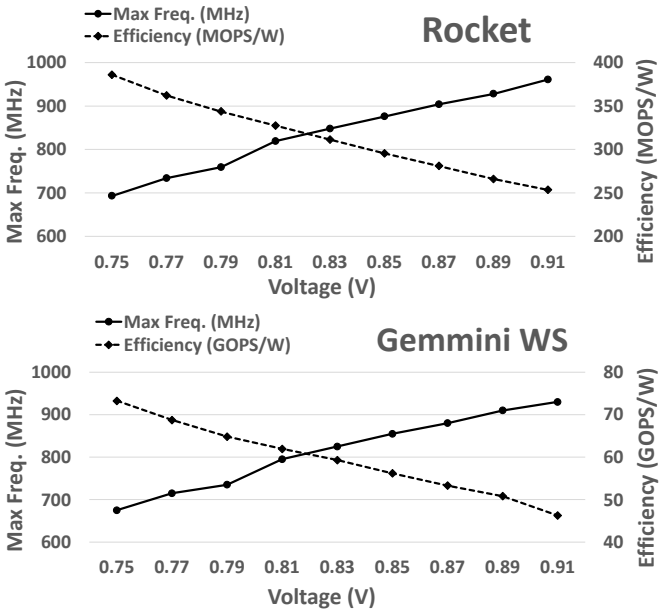

Fig. 7: Frequency/efficiency of the ML Gemmini-based domain.

Core Type		CoreMark Raw	CoreMark CM/MHz	Dhrystone Dhry/S	Dhrystone DMIPS/MHz
Rocket		1185	2.11	1883.54	1.07
BOOM	**Non spec.**	45.09	0.27	301.29	0.17
	Full	395.79	2.37	3750.93	2.13

TABLE II: CoreMark and Dhrystone results of CPUs (Rocket at 500MHz and BOOM at 167MHz).

L1 caches. For CoreMark/MHz, Rocket is 7.8x faster than the non-speculative BOOM core, while the full BOOM core is 8.8x faster reaching a peak of 2.37 CoreMark/MHz. For Dhrystone MIPS/MHz (DMIPS/MHz), Rocket is 6.3x faster compared to the non-speculative BOOM core, while the full BOOM core is 12.5x faster. Results show the flexibility of running software on the different cores and modes depending on the power, security, and performance requirements. Finally, unlike similar published test chips, this work contains a range of heterogeneous high-performance and energy-efficient cores and accelerators all while being fully open-sourced throughout the development process, as demonstrated in Table I.

V. Conclusion

This work demonstrates an open-source, heterogeneous multi-core multi-accelerator RISC-V SoC targeting general-purpose compute, DNN, and vector workloads. The 16mm^2 low-power Intel 22FFL test chip is integrated with fully open-source components including a BOOMv2.2 out-of-order core with re-configurable performance gating, a Hwacha vector accelerator, a Rocket in-order core, a Gemmini systolic array DNN accelerator, 1MiB of L2 cache, and a variety of off-chip I/Os. The wide amount of compute and variety of programmability on a single SoC allows for up to a 286x improvement in MOPS/W or 282x MOPS improvement over the RISC-V Rocket in-order core.

Acknowledgements

This research was supported by the DARPA CRAFT grant HR0011-16-C0052, DARPA Hi3 program, the ADEPT Intel ISTC on Agile Design, and ADEPT Lab industrial sponsors and affiliates.

References

[1] C. Schmidt *et al.*, "An Eight-Core 1.44GHz RISC-V Vector Machine in 16nm FinFET," *ISSCC*, 2021.

[2] F. Zaruba *et al.*, "The cost of application-class processing: Energy and performance analysis of a Linux-ready 1.7-GHz 64-bit RISC-V core in 22-nm FDSOI technology," *IEEE TVLSI*, vol. 27, no. 11, pp. 2629–2640, 2019.

[3] P. Chiu *et al.*, "An Out-of-Order RISC-V Processor with Resilient Low-Voltage Operation in 28NM CMOS," in *VLSI*, 2018, pp. 61–62.

[4] C. Celio *et al.*, "The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-167*, 2015.

[5] Y. Lee *et al.*, "Hwacha Preliminary Evaluation Results, Version 3.8." *UC Berkeley, Tech. Rep. UCB/EECS-2015-264*, 2015.

[6] K. Asanovic *et al.*, "The Rocket Chip generator," *UC Berkeley, Tech. Rep. UCB/EECS-2016-17*, 2016.

[7] H. Genc *et al.*, "Gemmini: An Agile Systolic Array Generator Enabling Systematic Evaluations of Deep-Learning Architectures," *arXiv preprint arXiv:1911.09925*, 2019.

[8] A. Amid *et al.*, "Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, 2020.

[9] A. Izraelevitz *et al.*, "Reusability is FIRRTL ground: Hardware construction languages, compiler frameworks, and transformations," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 209–216.

[10] S. Karandikar *et al.*, "FireSim: FPGA-accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 29–42.

[11] P. Whatmough *et al.*, "A 16nm 25mm 2 SoC with a 54.5x Flexibility-Efficiency Range from Dual-Core Arm Cortex-A53 to eFPGA and Cache-Coherent Accelerators," *VLSI*, pp. C34–C35, 2019.