CIFER: A Cache-Coherent 12-nm 16-mm^2 SoC With Four 64-Bit RISC-V Application Cores, 18 32-Bit RISC-V Compute Cores, and a 1541 LUT6/mm^2 Synthesizable eFPGA

Ang Li, Ting-Jung Chang, Fei Gao, Tuan Ta, Georgios Tziantzioulis, Yanghui Ou, Moyang Wang, Jinzheng Tu, Kaifeng Xu, Paul Jackson, August Ning, Grigory Chirkov, Marcelo Orenes-Vera, Shady Agwa, Xiaoyu Yan, Eric Tang, Jonathan Balkind, *Member, IEEE*, Christopher Batten, *Member, IEEE*, and David Wentzlaff, *Member, IEEE*

***Abstract*—This letter presents CIFER, the world's first open-source, fully cache-coherent, heterogeneous many-core, CPU-FPGA system-on-chips. The 12 nm, 16-mm^2 chip integrates four 64-bit, OS-capable, RISC-V application cores; three TinyCore clusters that each contain six 32-bit, RISC-V compute cores (18 in total); and an electronic design automation-synthesized, standard-cell-based eFPGA. CIFER enables the decomposition of real-world applications and tailored execution (parallelization or specialization) per decomposed task. Our evaluation shows that: 1) the TinyCore clusters increase the throughput and energy efficiency of data- and thread-parallel tasks by up to 7.95× and 7.75× over one 64-bit core, respectively; 2) the eFPGA increases the throughput and energy efficiency of hardware-accelerable tasks by up to 9.29× and 10.62×, respectively; and 3) using coherent caches for data transfer between the processors and the eFPGA increases the throughput and energy efficiency by up to 11.1× and 10.5×, respectively.**

***Index Terms*—Cache memory, computer architecture, parallel architectures, programmable logic arrays, reconfigurable architectures, system-on-chip (SoC).**

I. Introduction

The drive for performance and energy efficiency in the post-Moore era has given rise to hardware acceleration and heterogeneous integration. However, the high design cost and programming complexity impede the broad adoption of heterogeneous system-on-chips (SoC).

This work presents *CIFER* [1] (Fig. 1), *the world's first open-source, fully cache-coherent, heterogeneous many-core, CPU-FPGA SoC*. By integrating OS-capable processors, parallel compute cores, and an embedded FPGA (eFPGA), CIFER enables efficient execution of various workloads across the parallelism-specialization spectrum.

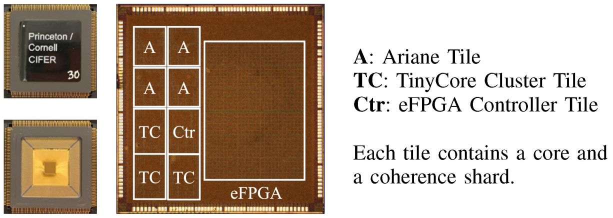

Princeton / Cornell CIFER
30
A
A
A
A
TC
Ctr
TC
TC
eFPGA
A: Ariane Tile
TC: TinyCore Cluster Tile
Ctr: eFPGA Controller Tile
Each tile contains a core and a coherence shard.

Fig. 1. CIFER package and die photos.

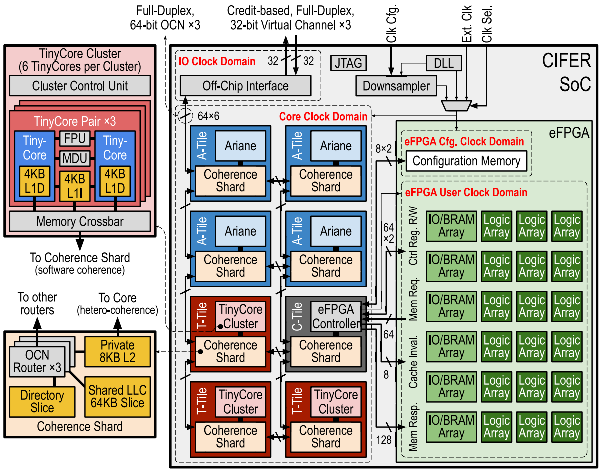


Fig. 2. CIFER SoC architecture.

CIFER lowers the design cost and the programming barrier with the following novelties. First, it demonstrates agile hardware development facilitated by open-source hardware. CIFER was designed in seven months during the pandemic by a team of graduate students and postdocs collaborating across two institutions, due in part to the use of many open-source projects, including OpenPiton [2], BYOC [3], PyMTL3 [4], PyOCN [5], Ariane [6], and PRGA [7]. Second, the eFPGA is synthesized with off-the-shelf electronic design automation (EDA) tools and standard cell libraries. Compared to the conventional, full-custom FPGAs, CIFER's synthesizable eFPGA is customizable in architecture, technology-agnostic, and flexible in physical layout. Third, CIFER implements different cache coherence schemes that are optimal for each processing unit and unifies them within a global, bi-directionally coherent cache system.

II. Architecture

The CIFER architecture (Fig. 2) integrates a 2×4 mesh of tiles and an eFPGA into the distributed, coherent, OpenPiton [2] P-Mesh

Manuscript received 9 July 2023; accepted 3 August 2023. Date of publication 8 August 2023; date of current version 7 September 2023. This work was supported in part by the Air Force Research Laboratory (AFRL) and in part by the Defense Advanced Research Projects Agency (DARPA) under Agreement FA8650-18-2-7852. This article was approved by Associate Editor Ningyuan Cao. *(Corresponding author: Ang Li.)*

Ang Li, Fei Gao, Georgios Tziantzioulis, Jinzheng Tu, Kaifeng Xu, Paul Jackson, August Ning, Grigory Chirkov, and David Wentzlaff are with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: angl@princeton.edu).

Ting-Jung Chang was with the Department of Computer Science, Princeton University, Princeton, NJ 08544 USA. She is now with the Miin Wu School of Computing, National Cheng Kung University, Tainan 701, Taiwan.

Tuan Ta, Yanghui Ou, Moyang Wang, Xiaoyu Yan, Eric Tang, and Christopher Batten are with the Department of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14850 USA.

Marcelo Orenes-Vera is with the Department of Computer Science, Princeton University, Princeton, NJ 08544 USA.

Shady Agwa was with the Department of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14850 USA. He is now with the School of Engineering, The University of Edinburgh, EH8 9YL Edinburgh, U.K.

Jonathan Balkind is with the Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA 93106 USA.

Digital Object Identifier 10.1109/LSSC.2023.3303111

cache system over three packet-switched, on-chip networks (OCNs) designed with PyOCN [5]. Each tile consists of a shard of the coherence system and one of the following: an Ariane core, a TinyCore cluster, or an eFPGA controller. Each coherence shard contains a private, 8 kB, L2 cache and a 64-kB slice of the shared, 512 kB, last-level cache (LLC). Coherence between the L2 caches and the LLC is maintained in hardware with a directory-based MESI protocol.

A. Ariane: OS-Capable Processor

Ariane [6] is an OS-capable, 64-bit, RISC-V processor with a six-stage in-order pipeline, a 16-kB L1 instruction (L1I) cache, an 8-kB L1 data (L1D) cache, and a double-precision floating-point unit (FPU). Coherence between Ariane's L1 caches and the L2 cache is maintained in hardware through adaptation to BYOC's transaction response interface (TRI) [3]. CIFER is the first silicon instantiation of BYOC.

B. TinyCore Cluster: Thread-Level Parallel Array

Each TinyCore cluster contains six 32-bit, RISC-V cores organized into three pairs. The six cores use an MIMD execution model, where each core executes an independent stream of instructions. Each core has a six-stage, in-order issue, out-of-order write-back, late-commit, and scalar pipeline. To address write-after-write and write-after-read hazards during out-of-order execution, each core supports limited register renaming with more physical registers (40 integers and 40 floating-point) than the 32 architectural registers specified in the RISC-V ISA.

Each core has a private, 4-kB L1D cache, while a pair of cores share a 4-kB L1I cache, an integer multiply-divide unit (MDU), and a single-precision FPU. A small L0 instruction buffer is added to each core's front-end to minimize the latency impact of sharing the L1I cache. Coherence between the L1D caches and the L2 cache is managed explicitly in software by inserting special cache flush and invalidation instructions. In particular, a cache flush traverses the L1D cache to write back each dirty cache line, while a cache invalidation clears the valid bits of the clean cache lines. Cache invalidation requests from the L2 cache are not propagated to the L1D caches. Sharing long-latency arithmetic units and reducing coherence hardware maximize computation density in each cluster.

C. Embedded FPGA: Reconfigurable Hardware Accelerator

The eFPGA (Fig. 3) is designed with PRGA [7]. It has 6720 multimode LUT6s and 18 24 kbit, dual-port, block RAMs (BRAMs). Hard-wired adder/carry chains are used for efficient emulation of arithmetic operations. The BRAMs support different word sizes, e.g., 512 × 48b, 1024 × 24b, ..., 24K × 1b. eFPGA-emulated, "soft" accelerators can be built with an open-source, RTL-to-bitstream toolchain consisting of Yosys [8], VPR [9], and PRGA's bitstream assembler.

The eFPGA contains three key novelties: First, the switch blocks implement a cycle-free connection pattern [10], facilitating automated, constraint-driven, area and timing optimization at the array level using off-the-shelf EDA tools. In comparison to the conventional FPGA design flow in which locally optimized blocks are tessellated in a predefined grid, this approach improves the power, performance, and area (PPA) of the synthesized FPGA by letting the EDA tools explore a larger design space. Second, the eFPGA is designed as a three-level hierarchy to balance PPA optimization and EDA runtime. The eFPGA is partitioned into two types of subarrays, namely, logic array and IO/BRAM array, which are then composed of logic blocks. Third, the eFPGA uses a hierarchical configuration network clocked by a multisource clock mesh. This enables fast and partial reconfiguration of the eFPGA at GHz clock frequency. In particular, each subarray contains a bitstream router and a single-bit scanchain that connects all the configuration cells with minimum routing metal usage. Bitstream segments are first sent to the bitstream routers over an 8-bit, packet-switched network, then buffered and shifted into the scanchains.

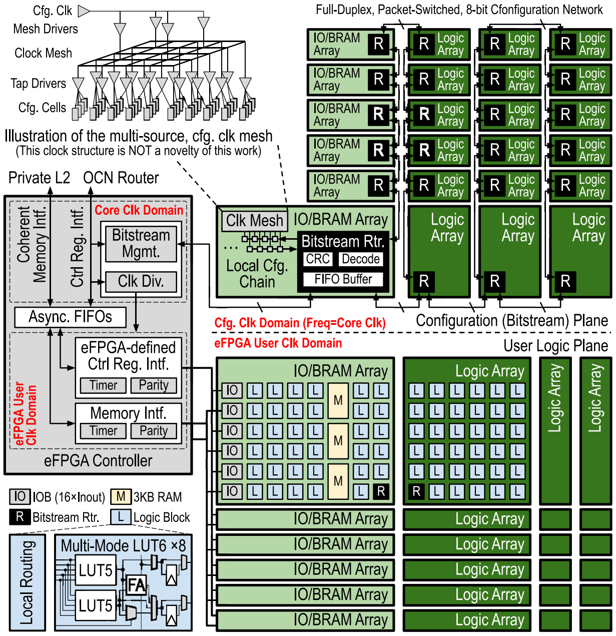


Fig. 3. CIFER eFPGA architecture.

The eFPGA is integrated with the system through the eFPGA controller, the first silicon instantiation of Duet [11], which contains the following two interfaces. The control register interface allows the processors to access the eFPGA via memory-mapped I/O. The coherent memory interface is configurable at runtime to enable non-coherent, IO-coherent, or bi-directionally coherent memory accesses of the eFPGA. In bi-directionally coherent mode, cache invalidation requests from the L2 cache are forwarded into the eFPGA, allowing the accelerator to include a soft cache. Atomic requests from the eFPGA are also supported, enabling low-overhead synchronization in user mode. Both interfaces contain asynchronous FIFOs for clock domain crossing and are equipped with timers and parity checks to protect the OCN and the memory system from software or accelerator bugs.

D. Heterogeneous Cache Coherence

One key contribution of CIFER is that it unifies the heterogeneous cache coherence schemes of each processing unit within a global, fully coherent cache system. This minimizes the communication overhead and maximizes the programmability of the SoC. For example, a task-parallel work-stealing runtime [12] facilitates parallel execution across the Ariane cores and the TinyCore clusters, leveraging the coherent caches and automating the insertion of cache management instructions. An eFPGA-emulated accelerator can be efficiently invoked by passing the memory addresses of the data to be processed. Depending on the computation, the accelerator can either copy a continuous chunk of data into its BRAM scratchpad or read/write memory in a random, byte-granular manner. This saves CPU cycles from explicitly managing data movement and prevents over-fetching from the eFPGA.

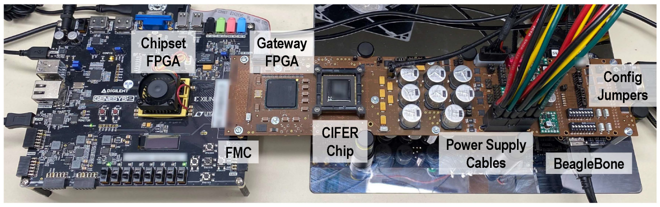


Fig. 4. Lab evaluation setup.

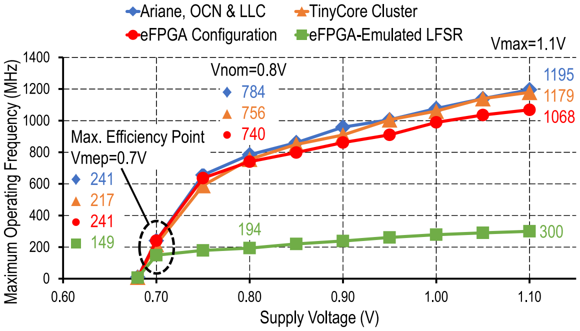
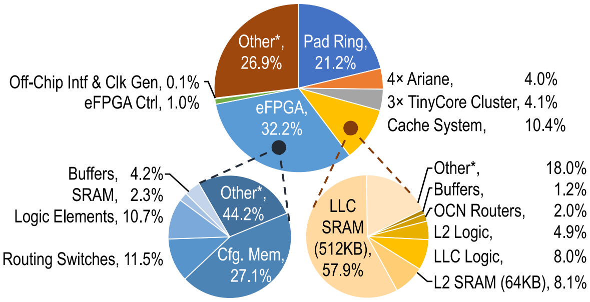
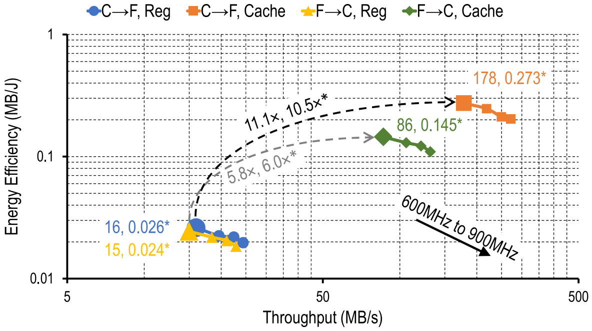


Fig. 5. Max. operating frequency versus supply voltage.



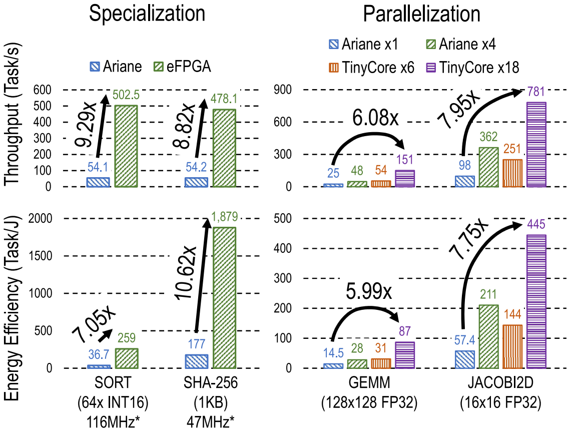
* **Other** includes physical cells (e.g., tap cells, decap cells, boundary cells), gap areas between hard macro blocks, etc.

Fig. 6. Area breakdown.



* Throughput and energy efficiency when the processors run at 600MHz. The eFPGA runs at 1/16 of the CPUs' clock frequency. C=CPU; **F**=eFPGA; **Reg**=memory-mapped I/O; **Cache**=coherent cache.

Fig. 7. CPU-FPGA communication throughput and energy efficiency at different system clock frequency.



* F_{max} of the eFPGA-emulated accelerator. The processors, OCN, cache system, and the eFPGA controller runs at full speed (740MHz at 0.8V).

Fig. 8. Performance and efficiency gains from offloading.

III. Evaluation

Fig. 4 shows our chip testing setup. Fig. 5 shows each component's maximum operating frequency (F_{max}) across the range of functional supply voltages. The eFPGA's F_{max} depends on the emulated design, and Fig. 5 shows the F_{max} of a 64-bit LFSR. Fig. 6 shows the area breakdown of the chip. The eFPGA's logic and routing resources only make up a quarter of the eFPGA's total area, while the configuration memory consumes another quarter. The eFPGA's low area utilization is due in part to the hierarchical design and can be improved with abutted or narrow-channel macro-placement strategies.

Table I compares CIFER with other state-of-the-art CPU-FPGA SoCs targeting the edge/IoT domain. Due to tooling issues, we did not implement explicit clock-gating on the eFPGA's configuration clock, which should be disabled except when loading the bitstream. Post-layout power analysis shows that the configuration clock subtree consumes about 90% of the chip's total clock power due to the high total capacitance and short-circuit current of the clock meshes. We estimate the total power with proper clock gating by subtracting the analyzed configuration clock power from the measured total power. Estimated numbers are shown in brackets, next to their measured counterparts in the table.

CIFER runs up to 1195 MHz at 1.1 V. The processors provide high aggregate performance with good energy efficiency, totaling 15.54 GFLOPS at 1.1 V and 53.18 GFLOPS/W (estimated as explained above) at 0.7 V, outperforming the next best SoC by 8.0× and 1.4×. The eFPGA's area efficiency is 1541 LUT6/mm^2, outperforming the other synthesizable eFPGAs by 11.2×, and is only 1.3× worse than the best full-custom eFPGA. The eFPGA's peak performance (1.92 MOPS/LUT, 126 MHz at 1.1 V) and energy efficiency (148.1 GOPS/W at 0.7 V) are measured with a 64-point FFT that utilizes 97% of the logic blocks and 75% of the BRAMs. The 3.4× performance gap and the 2.1× energy efficiency gap between the full-custom eFPGA and this work can be attributed to three factors: 1) CIFER is synthesized with standard cells; 2) our eFPGA has no hardware multiply-accumulate units; and 3) this work uses an open-source RTL-to-bitstream toolchain.

Fig. 7 shows the throughput improvements and energy savings when data are transferred through the coherent caches instead of memory-mapped I/O. The improvements are due to two reasons: 1) memory-mapped I/O accesses are strictly serialized in the processor's pipeline, while coherent caches may hide the latency of consecutive memory accesses, e.g., by buffering memory requests in the asynchronous FIFOs and 2) the eFPGA can use the L2 cache co-located in the eFPGA controller tile which runs in the fast, processors' clock domain.

Fig. 8 shows the throughput and energy efficiency gains by offloading four representative edge applications to their preferred compute unit. SORT and SHA-256 use eFPGA-emulated accelerators, while GEMM and JACOBI2D use the TinyCore clusters. The execution time is measured from when an Ariane core initiates a task to when

TABLE I
COMPARISON TO THE STATE OF THE ART

				This Work	TCAS'20 [13]	ISSCC'19 [14]	TVLSI'21 [15]	JSSC'22 [16]
Chip		Technology		12nm FinFET	90nm BCD	40nm CMOS + 39nm MRAM	22nm FD-SOI	16nm FinFET
		Die Area (mm^2)		16	1.78	22.09	9	25
		V_{nom} (V_{min} - V_{max})		0.8 (0.68 - 1.1)	1.2 (-)	- (1.1 - 1.3)	0.8 (0.5 - 0.8)	0.8 (0.5 - 1.05)
		Active Power (mW)	V_{nom}	1792	1.2	5.34	24.95	918
		F_{max} (MHz)	V_{max}	**1195**	10	200	600	972
CPU	Host	Core Type		4× Ariane	RI5CY	Cortex-M0	RI5CY	2× Cortex-A53
		ISA		RV64GC	RV32I	ARMv6-M	RV32IMFC	ARMv8-A
		CoreMark Score	V_{max}	**7918**	31.9	466	1914	6376
	Other	Core Type		18× TinyCore	N/A	N/A	N/A	Cortex-M0
		ISA		RV32IMAF				ARMv6-M
		Function		Parallel Compute				Monitor
		CoreMark Score	V_{max}	**19198**				2265
	Total	Peak GFLOPS	V_{max}	**15.54**	NO HW FPU	NO HW FPU	NO HW FPU	1.94
		Peak GFLOPS/W	V_{mep}	6.63 [**53.18†**]				38.03
eFPGA		IP		**Synthesizable w/ Std. Cells**	Synthesizable w/ Std. Cells	Unknown	Full-Custom Hard Macro	Full-Custom Hard Macro
		Min. Prog. Time (μs)		**239.4 - 1274.8**	-	-	-	450
		LUT Type & Count		6720 LUT6	48 LUT6	1176 LUT6	6000 LUT4	8760 LUT6
		Logic Density (LUT/mm^2)		1541	137	36	1505	1991
		F_{max} (MHz)	V_{max}	300**	1.25	200	193	747
		MOPS/LUT	V_{max}	1.92‡ (INT8)	-	-	0.02 (INT32)	6.45 (INT8)
		GOPS/W	V_{mep}	148.1$^{\ddagger *}$ (INT8)	-	-	29.1 (INT32)	312.4 (INT8)
Shared Memory BW (MB/s) [V_{max}]			CPU → eFPGA	**201**	Non-Coherent	Non-Coherent	Non-Coherent	-
			eFPGA → CPU	**558**				486

† Estimated power dissipation, excluding the eFPGA's configuration clock power based on post-layout power analysis
** Measured when the eFPGA emulates a 64-bit LFSR
‡ Measured when the eFPGA emulates an INT8-precision, complex, 64-point FFT
* Measured power dissipation in the eFPGA's user clock domain

the same core reads back all the results. All the control overhead is included, while the data transfer overhead is mitigated by overlapping compute with ad hoc, coherent memory accesses. At nominal voltage (0.8 V), the eFPGA outperforms the Ariane-only baseline by up to 9.29× in throughput and 10.62× in energy efficiency; the TinyCore clusters improve the performance and energy efficiency by up to 7.95× and 7.75×, respectively.

IV. CONCLUSION

This letter presents CIFER. Through cache-coherent integration of OS-capable processors, parallel many-core arrays, and an eFPGA, CIFER improves performance and energy efficiency on a wide range of workloads across the parallelism-specialization spectrum. The heterogeneous cache coherence scheme minimizes communication overhead and maximizes the programmability of the SoC. The EDA-synthesized, standard-cell-based eFPGA's area efficiency, peak performance, and energy efficiency are approaching those of full-custom eFPGAs.

ACKNOWLEDGMENT

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

REFERENCES

[1] T.-J. Chang et al., "CIFER: A 12nm, 16mm^2, 22-core SoC with a 1541 LUT6/mm^2 1.92 MOPS/LUT, fully synthesizable, cachecoherent, embedded FPGA," in *Proc. CICC*, pp. 1–2.

[2] J. Balkind et al., "OpenPiton: An open source manycore research framework," in *Proc. ASPLOS*, pp. 217–232.

[3] J. Balkind et al., "BYOC: A "bring your own core" framework for heterogeneous-ISA research," in *Proc. ASPLOS*, pp. 699–714.

[4] S. Jiang, B. Ilbeyi, and C. Batten, "Mamba: Closing the performance gap in productive hardware development frameworks," in *Proc. DAC*, pp. 1–6.

[5] C. Tan et al., "PyOCN: A unified framework for modeling, testing, and evaluating on-chip networks," in *Proc. ICCD*, pp. 437–445.

[6] F. Zaruba and L. Benini, "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 Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2629–2640, Nov. 2019.

[7] A. Li and D. Wentzlaff, "PRGA: An open-source FPGA research and prototyping framework," in *Proc. FPGA*, pp. 127–137.

[8] C. Wolf. "Yosys open SYnthesis suite." 2020. [Online]. Available: https://yosyshq.net/yosys/

[9] K. E. Murray et al., "VTR 8: High-performance CAD and customizable FPGA architecture modelling," *ACM TRETS*, vol. 13, no. 2, p. 9, 2020.

[10] A. Li, T.-J. Chang, and D. Wentzlaff, "Automated design of FPGAs facilitated by cycle-free routing," in *Proc. FPL*, pp. 208–213.

[11] A. Li, A. Ning, and D. Wentzlaff, "Duet: Creating harmony between processors and embedded FPGAs," in *Proc. HPCA*, pp. 745–758.

[12] M. Wang, T. Ta, L. Cheng, and C. Batten, "Efficiently supporting dynamic task parallelism on heterogeneous cache-coherent systems," in *Proc. ISCA*, pp. 173–186.

[13] F. Renzini, C. Mucci, D. Rossi, E. F. Scarselli, and R. Canegallo, "A fully programmable eFPGA-augmented SoC for smart power applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 2, pp. 489–501, Feb. 2020.

[14] M. Natsui et al., "12.1 an FPGA-accelerated fully nonvolatile microcontroller unit for sensor-node applications in 40nm CMOS/MTJ-hybrid technology achieving 47.14μW operation at 200MHz," in *Proc. ISSCC*, pp. 202–204.

[15] P. D. Schiavone et al., "Arnold: An eFPGA-augmented RISC-V SoC for flexible and low-power IoT end nodes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 4, pp. 677–690, Apr. 2021.

[16] S. K. Lee, P. N. Whatmough, M. Donato, G. G. Ko, D. Brooks, and G.-Y. Wei, "SMIV: A 16-nm 25-mm^2 SoC for IoT with arm cortex-A53, eFPGA, and coherent accelerators," *IEEE J. Solid-State Circuits*, vol. 57, no. 2, pp. 639–650, Feb. 2022.