The Rise and Fall of Automatic Instruction-Set Extensions

Paolo Ienne
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, CH
paolo.ienne@epfl.ch

Laura Pozzi
USI Lugano
6900 Lugano, CH
laura.pozzi@usi.ch

Philip Brisk
University of California Riverside
Riverside, CA 92521, US
philip@cs.ucr.edu

Customizable processors—which allowed their *instruction set architecture (ISA)* to be augmented with application-specific custom *instruction set extensions (ISEs)*—arrived on the market at the turn of the millenium. Commercial offerings included the Tensilica Xtensa [1], ARC ARCtangent [2], STMicroelectronics ST200 [3] and MIPS with CorExtend [4]. Academic researchers developed compiler techniques to extract application-specific ISEs directly from high-level software source code, synthesize them in hardware, and integrate them into an extensible ISA. An early example of this work was a paper published by two of the authors in the *Proceedings of the 14th International Conference on Application-specific Systems, Architectures and Processors (ASAP)* in 2003 [5].

This short retrospective looks back at some of the work of those years, reflects on why the topic all but disappeared from the research scene without producing any lasting industrial impact, and wonders about persisting threads of these efforts that may exist within the RISC-V ecosystem.

I. What are Instruction-Set Extensions?

Following the RISC vs. CISC debate of the 1980s, the 1990s marked a renewed interest in the design of instruction sets, ranging from general-purpose [6] to application-specific [7]. A middle ground arose, which was to extend a Turing-complete, often RISC-like, instruction set with application-specific ISEs to accelerate performance-critical kernels (e.g., [8]). The standard core guarantees the availability of good compilers, while a small number of well-chosen ISEs can dramatically improve the performance of targeted applications.

II. Why Automatic?

In the 1980s and early 1990s, successful academic *High-Level Synthesis (HLS)* projects suggested that a pathway from high-level languages to hardware was within reach. Yet, Synopsys strategically introduced *Behavioral Compiler* in 1994, only to discontinue it within a decade. Automated processor customization emerged as an alternative way forward. Chris Rowen, Tensilica's founder, used the slogan "The processor is the NAND gate of the future." *Electronic Design Automation (EDA)* researchers and companies saw an opportunity to build a new synthesis ecosystem around customizable processors. Augmenting traditional compilers with algorithms for ISE identification on one side, and HLS for ISE hardware on the other, offered a pathway to compile applications written in imperative programming languages to custom silicon [9].

III. The 2000s: Academic Research on ISEs

Galuzzi and Bertels provide a comprehensive survey that covers the majority of impactful compiler techniques for algorithmic ISE identification through 2011 [10]. Many early papers focused on maximizing ISE reuse across an application. As a counterpoint, the paper at the center of this retrospective applied symbolic algebra to mathematically decompose application-level polynomials into candidate ISEs [5], emphasizing a more fundamental—if unconventional—form of reuse, focusing on ISEs that are dynamically executed frequently and thereby maximize speedup.

Two of our seminal works on automatically extracting ISEs [11], [12] under realistic microarchitectural constraints have had some impact; the citation trend over the years, shown in Figure 1, provides a visual narrative of this research area: initial hype in the 2000s and early 2010s, followed by (inevitable?) decline. Subsequent work explored a variety of topics, including multi-cycle ISEs [13], incorporation of architecturally visible storage [14], and handling simple control flow structures [15], [16]. By the late-2000s, it became clear to academic researchers that ISE performance was limited by the ability of the CPU microarchitecture to supply data to the ISE hardware; the workaround was to allow ISEs to access memory directly [17], [18]. To the best of our knowledge, most of these innovations have not been adopted by industry.

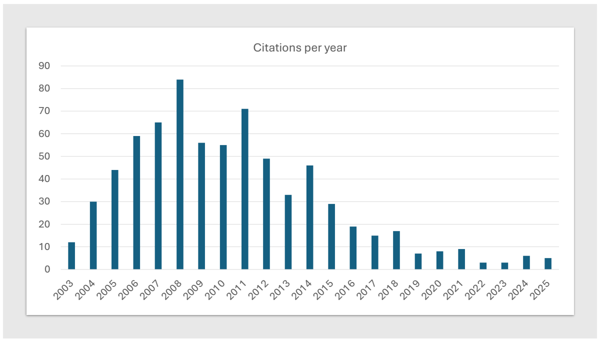

Fig. 1. Citation counts of our two seminal papers [11], [12] (Google Scholar).

IV. The 2010s: Industry Consolidation

The era preceding the current AI boom saw Cadence Design Systems and Synopsys respectively purchasing Tensilica and ARC; the technology remains available today as commercial IP. The development of HLS continued to evolve, with increasingly sophisticated algorithms for scheduling complex loops [19], and emerged as a viable compilation strategy for FPGAs. In 2011, Xilinx acquired HLS startup AutoESL, and integrated their technology as Vivado HLS, which is now part of Vitis HLS. Many concepts germane to ISEs were kept alive through research on decoupled programmable FPGA-based vector accelerators, which could be controlled by a hard or soft CPU [20]; one startup company in this area, VectorBlox, was purchased by Microchip in 2019, and the technology remains commercially available today as a software development kit.

V. Hardware Compilers for Programmers?

To software programmers, compilers are invisible tools—they just work. Hardware generation is a different story: HLS often demands significant code refactoring, and integrating the generated hardware remains tightly coupled to vendor-specific toolchains and platforms. An underappreciated advantage of automatic ISE generation was its high degree of automation: with little to no manual intervention, it could transparently produce custom instructions that seamlessly fit into existing compilation flows and require minimal system integration effort. Will HLS tools ever achieve this level of transparency? Or are ISEs the right, more powerful abstraction to realize it?

VI. The 2020s: the RISC-V Ecosystem

RISC-V, an open standard ISA, is extensible by design [21] and represents an opportunity to revive both academic and commercial interest in ISEs. At present, numerous entities, both academic and commercial, have created bespoke RISC-V CPUs with custom ISEs, amidst a wide variety of vendor tooling options for doing so. One company operating in the vendor tooling space is a European startup, Codasip [22], whose software solutions seem similar to what Tensilica offered 20+ years ago. However, a Balkanized ecosystem in support of a standard creates challenges such as interoperability and backwards compatibility; to address these challenges, several individuals have proposed a specification for RISC-V composable custom extensions [23], which may be adopted in the future. If these efforts are successful, it seems possible that one or more vendors may emerge with commercial tools that automatically identify and synthesize ISEs for RISC-V CPUs, possibly based on algorithms derived from academic work on the topic published by the authors, and others, in the 2000s.

References

[1] T. R. Halfhill, "Tensilica's software makes hardware," *Microprocessor Report*, 23 Jun. 2003.

[2] ——, "ARC Cores encourages "plug-ins"," *Microprocessor Report*, 19 Jun. 2000.

[3] P. Faraboschi, G. Brown, J. A. Fisher, G. Desoli, and F. Homewood, "Lx: A technology platform for customizable VLIW embedded processing," in *Proceedings of the 27th Annual International Symposium on Computer Architecture*, Vancouver, Jun. 2000, pp. 203–13.

[4] T. R. Halfhill, "MIPS embraces configurable technology," *Microprocessor Report*, 3 Mar. 2003.

[5] A. Peymandoust, L. Pozzi, P. Ienne, and G. D. Micheli, "Automatic instruction set extension and utilization for embedded processors," in *Proceedings of the 14th International Conference on Application-specific Systems, Architectures and Processors*, The Hague, The Netherlands, Jun. 2003, pp. 103–14.

[6] B. K. Holmer and A. M. Despain, "Viewing Instruction Set Design as an Optimization Problem," in *Proceedings of the 24th Annual International Symposium on Microarchitecture*, Albuquerque, N.Mex., 1991, pp. 153–162.

[7] I.-J. Huang and A. M. Despain, "Synthesis of application specific instruction sets," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-14, no. 6, pp. 663–75, Jun. 1995.

[8] H. Choi, J.-S. Kim, C.-W. Yoon, I.-C. Park, S. H. Hwang, and C.-M. Kyung, "Synthesis of application specific instructions for embedded DSP software," *IEEE Transactions on Computers*, vol. C-48, no. 6, pp. 603–14, Jun. 1999.

[9] G. Martin and G. Smith, "High-Level Synthesis: Past, present, and future," *IEEE Design and Test of Computers*, vol. 26, no. 4, pp. 18–24, Jul.–Aug. 2009.

[10] C. Galuzzi and K. Bertels, "The Instruction-Set Extension Problem: A Survey," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 4, no. 2, pp. 1–28, 2011.

[11] K. Atasu, L. Pozzi, and P. Ienne, "Automatic application-specific instruction-set extensions under microarchitectural constraints," in *Proceedings of the 40th Design Automation Conference*, Anaheim, Calif., Jun. 2003, pp. 256–61.

[12] L. Pozzi, K. Atasu, and P. Ienne, "Exact and approximate algorithms for the extension of embedded processor instruction sets," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-25, no. 7, pp. 1209–29, Jul. 2006.

[13] L. Pozzi and P. Ienne, "Exploiting pipelining to relax register-file port constraints of instruction-set extensions," in *Proceedings of the International Conference on Compilers, Architectures, and Synthesis for Embedded Systems*, San Francisco, Calif., Sep. 2005, pp. 2–10.

[14] P. Biswas, N. Dutt, P. Ienne, and L. Pozzi, "Automatic identification of application-specific functional units with architecturally visible storage," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Munich, Mar. 2006, pp. 212–17.

[15] M. Zuluaga, T. Kluter, P. Brisk, N. Topham, and P. Ienne, "Introducing control-flow inclusion to support pipelining in custom instruction set extensions," in *Proceedings of the 7th IEEE Symposium on Application Specific Processors*, San Francisco, Calif., Jul. 2009, pp. 114–21.

[16] G. Zacharopoulos, L. Ferretti, E. Giaquinta, G. Ansaloni, and L. Pozzi, "RegionSeeker: Automatically identifying and selecting accelerators from application source code," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 741–54, Apr. 2018.

[17] T. Kluter, P. Brisk, P. Ienne, and E. Charbon, "Speculative DMA for architecturally visible storage in instruction set extensions," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, Atlanta, Ga., Oct. 2008, pp. 243–48.

[18] T. Kluter, P. Brisk, E. Charbon, and P. Ienne, "Way stealing: A unified data cache and architecturally visible storage for instruction set extensions," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. VLSI-22, no. 1, pp. 62–75, Jan. 2014.

[19] J. Cong and Z. Zhang, "An efficient and versatile scheduling algorithm based on SDC formulation," in *Proceedings of the 43rd Design Automation Conference*, San Francisco, Calif., Jul. 2006, pp. 433–38.

[20] A. Severance and G. G. F. Lemieux, "Embedded supercomputing in FPGAs with the VectorBlox MXP Matrix Processor," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, 2013, pp. 1–10.

[21] *RISC-V Instruction Set Manual*, RISC-V, 2025, https://github.com/riscv/riscv-isa-manual.

[22] *Champions of Custom Compute*, Codasip, 2025, https://codasip.com/.

[23] *Composable Custom Extensions Specification*, Gray Research, 2024, https://github.com/grayresearch/CX/tree/main/spec.