RETROSPECTIVE: PipeRench: a Coprocessor for Streaming Multimedia Acceleration

Seth Copen Goldstein*, Herman Schmit†, Matthew Moe‡, Mihai Budiu§, Srihari Cadambi¶, R. Reed Taylor‖, Ron Laufer**
*School of Computer Science, Carnegie Mellon University, seth@cmu.edu
†Google DeepMind, schmit@google.com
‡PDF Solutions, mattmoe@gmail.com
§Feldera, mbudiu@feldera.com
¶Google, cadambi@google.com
‖Google, reedtaylor@google.com
**Keysight, ronald.laufer@gmail.com

I. Context for the Paper

In the early 1990s, academics were exploring alternatives to the Von Neumann stored-program architecture by combining high-level hardware synthesis with reprogrammable FPGAs. In the extreme, their vision was to replace a store of sequential instructions and a program counter with a single "instruction," essentially a description of the circuit that implemented an entire algorithm (with all its attendant parallelism). Changing the program could be accomplished by simply re-writing the FPGA configuration. Despite showing orders of magnitude speedup on data-intensive applications, this movement was largely dismissed by mainstream computer architecture researchers. In part, this was reasonable given the lack of high-level languages, the poor tool chains (AKA long "compile" times), poor system integration, and the fixed size of the target FPGAs with no easy way to take advantage of Moore's Law leading to larger FPGAs. It was also largely ignored by the FPGA companies whose revenue was mostly in logic emulation and the consolidation of random logic on printed circuit boards.

Starting around 1994, DARPA initiated a number of programs at universities and industry to investigate the applicability of so-called "adaptive computing" techniques. At Carnegie Mellon, our intent within this program was to investigate what could happen if you could change the configuration of an FPGA as fast as you change the instructions in a conventional computer. We sought to use "dynamic reconfiguration" to provide some of the things that were inadequate about the FPGA as a computing target. Specifically, we sought to:

- Accelerate compilation speed: Conventional FPGAs took hours to convert an algorithmic description to the bits required to program the device. This made the edit-compile-run loop comically unproductive. We hoped to make that a reality by "virtualizing" an FPGA and not making the compiler responsible for meeting every fixed constraint of the target device.
- Provide cross-generational value of software: Given the rapid pace of Moore's law, if a particular bitstream would only run on one instance of a chip in one process technology, it could never keep up with the exponential acceleration of old code experienced by CPUs. We hoped that by creating some independence between representation and execution, we could enable the investment in hardware configurations to pay off for multiple generations of chips.

Along the path to those goals, we discovered some things. First, we simply could not afford, in terms of power or wires, to change the entire FPGA configuration every cycle. If we could only configure a portion at a time, how could we split up the computation so that configuring a portion of the application every cycle made sense? If the computation was a pipeline, we could configure each stage as the pipeline filled with data. This is where the idea of pipeline configuration originated. Second, even with pipeline configuration, instruction bandwidth was still a problem. As a result, we had to reconsider the idea of a bit-level FPGA-like fabric. Using a larger processing element (PE) and bus size would reduce instruction bandwidth, but it would also lose some efficiency on the variable datawidth applications. The central evaluation in the ISCA paper is the trade-off between PE bit-size and computational density given a constrained instruction path.

In addition to the technical advances, the PipeRench paper updated the methodology used to investigate this field based on the methodology used in much of the ISCA community. We invested in building out a diverse set of application kernels, a compiler that could retarget different variants of the PipeRench architecture, and an area and performance model that would allow us to make the right trade-offs.

II. What Happened to PipeRench?

After this paper's publication, we successfully designed, fabricated, and got the PipeRench chip working in our lab [6]. It successfully ran our DCT, FIR filter, and IDEA encryption kernels faster than CPUs of the day. We used the baseline of this work to explore a set of different applications and alternative implementations [3], [4], accelerated compilation techniques. At least five students received PhDs based on extensions to the PipeRench chip, compiler, and architecture.

We attempted to commercialize the technology first with a Carnegie Mellon spin-off company. That company was too late for one Silicon Valley bubble, and too early for the next one. Later, CMU licensed the technology to two semiconductor companies in succession. But, it never became a commercial reality.

III. Influence of PipeRench

The idea of dynamic reconfiguration is now practically a requirement in modern FPGAs. Intel and AMD FPGAs both provide the ability to dynamically reconfigure a portion of their circuitry, while other portions continue to operate. This allows a portion of the FPGA to have a "privileged" hardware design, and a customer region that can be changed to support cloud customers.

A number of academic and commercial efforts have attempted to leverage the idea of very-high speed dynamic reconfiguration. Among those efforts are:

- Startup company Tabula had parts that reconfigured an FPGA-like device every cycle to enable much denser logic density. In a way, it was like pipeline reconfiguration where the logic for every cycle was broken into a pipeline of sub-cycles.
- NEC Laboratories America, Inc. designed and prototyped a PipeRench-inspired architecture called SimPLE specifically for accelerating logic simulation in the Electronic Design Automation space. SimPLE didn't use virtualization; rather it was a statically configured VLIW-like architecture similar to PipeRench's fabric. The large number of interconnected FPGA primitives coupled with high off-chip memory bandwidth enabled it to achieve speedups over conventional processors [1].
- Stanford developed Plasticine [5], which is also honored as an influential paper in this quarter decade of ISCA, has a Pattern Compute Unit that looks fairly similar to the PipeRench fabric, consisting of functional units and a pass register file. This academic chip has been commercialized by the company SambaNova.

IV. How Did Our Predictions Work Out?

Without making a claim of causality, the PipeRench paper did anticipate a number of trends that are clearly evident today:

- Accelerator integration: Accelerators maybe the most important way to accelerate computing in a time after Moore's Law [2]. FPGAs or other spatial accelerators exist in cloud services such as Azure and AWS.
- Coarse-grained unit integration in FPGAs: All modern FPGAs have some hardened support for dense numerical operations, including floating point. And in an inverted version of our prognostication that every CPU would have FPGA-like coprocessor, almost all new high-end FPGAs incorporate a multi-core CPU.
- Spatial computing: Many if not most supercomputers contain FPGAs. GPUs and TPUs use spatially distributed computing for extreme high-performance.
- Embarrassingly parallel workloads dominate: Some of the reviewer feedback from the paper attacked the workloads as too easy. But in our current age of graphics and ML accelerators, appropriately dealing with outrageous amounts of parallelism is the primary challenge.

But there were things that definitely did not turn out exactly as we imagined:

- Compilation time: It still takes ages to place-and-route an FPGA or a spatial accelerator. Today, teams building an FPGA-based accelerator contemplate the dimensions of programmability that they will need for an accelerator, and include the right level of dynamic programmability into their device. It is rare to see a custom hardware design for a single instance of an algorithm.
- Virtualized hardware: Bob Colwell told a panel at an FPGA conference that, as long as Moore's Law was in effect, CPUs accelerated all the software ever written for them by 1% per week. The idea that the FPGA was always going to be losing ground unless there was a way to port applications to newer, denser, faster chips, was the motivation behind PipeRench hardware virtualization. But the idea never really caught fire, and as Moore's law has slowed, it is less imperative. Software is now more open, which makes recompilation to a new target more practical, and binary executables do not carry the value they once did.

V. Final Thoughts

We are honored to be recognized by ISCA. It was a great opportunity to reconnect with our former colleagues, compare notes, and reflect the outcomes of this work that we were so passionate about.

References

[1] S. Cadambi, C. Mulpuri, and P. Ashar, "A fast, inexpensive and scalable hardware acceleration technique for functional simulation," in *Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324)*, 2002, pp. 570–575.

[2] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *Commun. ACM*, vol. 62, no. 2, p. 48–60, jan 2019. [Online]. Available: https://doi.org/10.1145/3282307

[3] H. Kagotani and H. Schmit, "Asynchronous piperench: architecture and performance evaluations," in *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003. FCCM 2003.*, 2003, pp. 121–129.

[4] R. Laufer, R. Taylor, and H. Schmit, "PCI-PipeRench and the SWORDAPI: a system for stream-based reconfigurable computing," in *Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines (Cat. No.PR00375)*, 1999, pp. 200–208.

[5] R. Prabhakar, Y. Zhang, D. Koeplinger, M. Feldman, T. Zhao, S. Hadjis, A. Pedram, C. Kozyrakis, and K. Olukotun, *Plasticine: A Reconfigurable Architecture For Parallel Paterns*. New York, NY, USA: Association for Computing Machinery, 2017, p. 389–402. [Online]. Available: https://doi.org/10.1145/3079856.3080256

[6] H. Schmit, D. Whelihan, A. Tsai, M. Moe, B. Levine, and R. Reed Taylor, "Piperench: A virtualized programmable datapath in 0.18 micron technology," in *Proceedings of the IEEE 2002 Custom Integrated Circuits Conference (Cat. No.02CH37285)*, 2002, pp. 63–66.