RETROSPECTIVE: NanoFabrics: Spatial Computing Using Molecular Electronics

Seth Copen Goldstein
School of Computer Science
Carnegie Mellon University
Email: seth@cmu.edu

Mihai Budiu
Feldera
mbudiu@feldera.com

I. Origins

The work in this paper is in large part an outgrowth of the DARPA Moletronics program which was focused on finding alternative manufacturing methods for computing devices. It brought together teams that spanned diverse fields across chemistry, materials science, physics, electrical engineering, and computer science. The PI meetings were incredible learning experiences and promoted out of the box thinking to somehow get beyond Moore's Law and create practical computing substrates that didn't depend on photolithography and silicon.

There were (and still are) many challenges to creating a computing device out of molecules: finding molecules with the right electrical properties, assembling the molecules into useful structures, handling the lack of precise placement and alignment, dealing with the intrinsic imperfections and defects that arise from the scalable methods being investigated, and on and on up the hierarchy. Interdisciplinary research is both rewarding and challenging. At one meeting we presented our approach towards creating a model of computation that would scale to exascale devices—the Split-phase Abstract Machine model—which we called the SAM model. Suddenly, as we discussed the SAM model, more people than usual were paying rapt attention. (It can be very hard to be continually attentive at interdisciplinary meetings when the hard and interesting problems in other fields are inaccessible.) What we thought to be a breakthrough in communication turned out to be a breakdown in acronyms. SAM to many experts in the room were "Self-Assembled Monolayers", an important method for tackling the problem of assembling the molecules into devices.

Miscommunications aside, we learned a tremendous amount from being thrown together to investigate what was and still is a very hard, but also very interesting problem—how to build practical computing devices from nanoscale devices using chemical assembly, AKA, Chemically Assembled Electronic Nanotechnology (CAEN). The main characteristic of CAEN from our perspective is that it uses directed self-assembly to assemble the individual components into a final product. This naturally leads to highly regular designs. It also leads to high defect rates. Reconfigurable computing becomes a natural target architecture for such a process since the information complexity of the desired circuit can be added after manufacturing and, as the Teramac project [1] showed, defects could be mapped and then avoided. What made the idea of CAEN-based reconfigurable fabrics so interesting, however, was that the molecular components being investigated could be changed *in situ* with a resulting change in their electrical properties, e.g., from a diode to an insulator. Thus, one of the main drawbacks to reconfigurable fabrics, the overhead of the configuration, could be eliminated in CAEN-based fabrics.

II. Scaling

The overriding theme of our research was scaling: at manufacturing time, at compilation, and at run-time. Our approach, influenced by our previous work on reconfigurable computing, was to make the manufacturing problem easier by narrowing the problem to creating regular arrays which could be configured, after manufacturing, into the circuits we desired. Even so, the novel nature of the underlying technology meant that most of the focus of the paper was in making a case that the architecture proposed architecture was possible. Twenty years later people are still doing research into array-based nanoscale architectures. While none use the specific devices we describe nor have they been commercialized, the general approach is still an active area of research. We think that credit for this should go to the ISCA program committee that was willing to include such a speculative piece of work in the program. The second piece of the manufacturing scalability problem is the built-in self-test needed to avoid the defects expected in such a device. This is also still an active area of research.

Scaling the compilation process to support the mapping of an entire program to the underlying reconfigurable fabric was the other main thrust of our paper. Our approach was to divide a program into a set of fixed latency basic-blocks, a sequence of instructions with a single entry-point and where every instruction had a fixed latency. Instructions with unknown latencies, e.g., memory operations, function calls ended a block. The blocks were each compiled into a circuit and the blocks communicated via an asynchronous protocol. This allowed us to decompose the compilation problem, to reduce constraints on the place-and-route problem, and to tolerate the latency of potentially unknown routes introduced by defects in the underlying fabric.

III. Spatial Computation

The Split-phase Abstract Machine we talked earlier became the basis for future work which we called spatial computation [2]. The limit study in the paper is based on instruction traces and makes some unrealistic assumptions. The most extreme assumption is that the memory addresses touched by a hardware instruction can be statically enumerated. We followed up on this work with a line of papers on spatial computation [2], [5], with more realistic assumptions. In spatial computation only computational resources are unlimited; memory is still centralized. Surprisingly, in spite of the numerous assumptions we made in this paper, the results from our actual compiled code were quite similar. We also investigated spatial computation using asynchronous self-timed circuits [6].

On thing we discovered is that even with unlimited computational resources, the compiler is unable to exploit parallelism in typical SPEC-INT programs to provide significant speed-ups due to control synchronization overheads. Spatial computation fabrics (e.g., FPGAs, GPUs, TPUs, etc.) are an attractive substrate for certain classes of parallel computations, with abundant explicit parallelism, in areas such as high-speed packet processing in Networking, computer graphics , machine learning , etc. In fact, today's problems seem particularly suited to this model and more than twenty years later we still find this idea attractive and worthy of further investigation.

IV. What Followed

One of the unintended consequences of working in this area was that Seth started thinking about reversing the use of the molecules involved. In the NanoFabric, we harnessed the conformational change of a molecule to alter it's I-V curve, creating a programmable diode/insulator. What if could change the shape of the molecule as a result of a computation? This was the seed behind Claytronics [4], a project Seth started with Todd Mowry on building and controlling programmable matter [3]. This project was also highly interdisciplinary including research into MEMS, power transmission, robotics, algorithms, language design, and compilers.

We want to thank DARPA and NSF for funding future looking research and the ISCA community for helping us develop the skills needed to think about designing and analyzing systems and how to think about, understand, and then harness the trends in technological change.

References

[1] R. Amerson, R. Carter, B. Culbertson *et al.*, "Teramac–configurable custom computing," in *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, D. A. Buell and K. L. Pocek, Eds., Napa, CA, Apr. 1995, pp. 32–38.

[2] M. Budiu, G. Venkataramani, T. Chelcea, and S. C. Goldstein, "Spatial computation," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Boston, MA, October 2004.

[3] S. C. Goldstein, J. D. Campbell, and T. C. Mowry, "Programmable matter," *Computer*, vol. 38, no. 6, pp. 99–101, 2005.

[4] S. C. Goldstein, T. C. Mowry, J. D. Campbell *et al.*, "Beyond audio and video: Using claytronics to enable pario," *AI Magazine*, vol. 30, no. 2, p. 29, Jun. 2009.

[5] M. Mishra, T. J. Callahan, T. Chelcea *et al.*, "Tartan: Evaluating spatial computation for whole program execution," in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XII, 2006, p. 163–174.

[6] G. Venkataramani, M. Budiu, and S. C. Goldstein, "C to asynchronous dataflow circuits: An end-to-end toolflow," in *International Workshop on Logic synthesis (IWLS)*, Temecula, CA, June 2004, pp. 501–508.