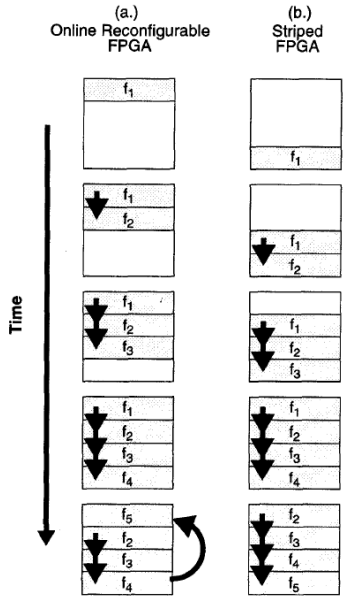
Incremental Reconfiguration for Pipelined Applications

Herman Schmit



Year of publication: *1997*
Area: *Run-time Systems and Run-Time Reconfiguration*

By the mid-1990s, FPGAs were being used for signal processing and computing. However, using FPGAs for computational tasks was hard. The FPGAs were small. An application developer had to be acutely aware of the FPGA capacity and massage the design to fit. The chip capacity created a performance cliff for designs. Furthermore, when a new, larger FPGA came along, it was necessary to redesign the application to exploit the new logic capacity. This was particularly unattractive to developers long accustomed to microprocessors, where you did not have to be aware of the size of your computation in order to get it working. Furthermore, once you had a design working, you could reasonably expect newer microprocessors to run the design faster without further development.

At the same time, FPGA users, vendors, and researchers were experimenting with runtime reconfiguration to create the illusion of additional logic capacity. While early runtime reconfiguration applications looked promising, they demanded more design effort and did not address the issue of scaling.

The signal processing and cryptography kernels that were showing good performance on FPGAs often obtained their performance benefits by exploiting pipeline parallelism---building a deep spatial pipeline for the computation. Schmit observed that pipelined computation could be used as an abstract model for these applications, and this model could be supported with a novel reconfiguration architecture to address the problem of design fit and scalability.

In particular, the pipeline provided a basis for loading only a small fraction of the configuration per cycle---the configuration for a single stage of the pipeline. It also served as a key unit of temporal locality---the same configuration could be reused on the next cycle to compute the next set of data flowing through the pipeline. The configuration could, itself, be pipelined through the computational fabric to spatially adjacent pipeline stages. This allowed (1) the reconfigurable array to be compact, holding a single configuration, (2) the configurations to live in large, dense memories outside of the array, and (3) the array to productively use limited bandwidth to the external configuration memory. The architecture could scale by adding physical pipeline stages.

This paper was the first of a series of papers about the architecture that would eventually be known as PipeRench. It identified the challenge and the basic solution, used simple analysis to show the potential benefits of the scheme, provided preliminary VLSI implementation characteristics, and illustrated support for a couple of applications. The PipeRench design later became a key part of the CMU Q-Machine and was briefly commercialized by Rapport, Inc.

André DeHon

DOI: http://dx.doi.org/10.1109/FPGA.1997.624604