

Not much has changed in computing in the last 50 years...

Most computers still use the von Neumann stored program model with serial execution of instructions, centralized main memory, and a bus tying it all together. We write algebraic formulas and symbolic constructs and surround them with various loop structures to sequentially process an array of data items. Language statements are translated into a fixed set of machine instructions which are then executed one (or perhaps two or three) at a time. Strange errors of unknown origin frequently occur, and even stranger incantations are often necessary to recover from them.

To be sure, the price/performance of computing hardware has improved dramatically over the years, on a curve of steepness and length that may never be equalled in any man-made technology. And computers have gotten dramatically easier to use, largely because user interfaces have become two-dimensional. Instead of typing arcane commands, we now merely push the mouse to a button displayed on the screen and click. Networking has added another significant dimension to computing applications.

But the way in which computers are built and programmed hasn't changed all that much. It can be argued that languages, compilers, CAD/CAE tools, etc. have improved somewhat, but *our underlying models of computation have remained basically unchanged since the earliest days of computing.*

The reasons for this are simple: Hardware is difficult and time-consuming to build and to change. Serial instruction execution is a simple, very general paradigm. Parallel architectures can be more powerful, but are less general. A special-purpose circuit can always outperform a microprocessor-based implementation (assuming the same integrated circuit technology is used for both) *for a small class of problems.* Usually the higher performance is achieved through parallelism, either spatial (replication) or temporal (pipelining). The very specialization which provides this parallelism also necessarily limits the range of its application.

I first became aware of a way in which things could be different in 1968, when I came upon Sven Wahlstrom's article in Electronics magazine entitled "Programmable Logic Arrays -- Cheaper by the Millions". Wahlstrom, Spandorfer, and other pioneers in this area had come up with a radical idea: instead of relying on permanent fuses or 'cutpoints' to customize an array of integrated circuitry, one could simply include additional gates to do the job. The result would be special-purpose hardware which was easily changed as well.

This idea was somewhat heretical when gates were precious, but most people knew this would not always be the case. I became sufficiently excited about this

direction to adopt it as my doctoral dissertation research at Carnegie-Mellon University, and completed the thesis in early 1970. At my thesis oral defense, one committee member asked how long it might be before programmable logic would be a reality and in common use in computing hardware. I thought it might be 5-10 years, given that the first microprocessor was already dimly visible in the integrated future. I suspended work in this area awaiting levels of integration which would make my thesis practical.

Graduate students are often known for their great optimism, but they are by no means unique in this regard. In fact it was nearly 15 years before even the first real FPGA was introduced into the marketplace. Today, 33 years after Wahlstrom's first patent was filed (1966), FPGAs are having a significant impact, but real reconfigurable computing still remains an elusive dream.

The real impact of FPGAs -- *restructurable computing* -- is yet to be felt. As many authors have duly pointed out, the ability to reconfigure hardware on the fly will have vast ramifications. Once suitable design tools and automatic methods become available, designers and programmers will be able to create custom hardware circuitry and pipelines to suit the problem at hand. I like the term "soft hardware", as it suggests that hardware will become as readily created and malleable as software. In a practical sense, this means that the turn-around time for custom hardware will be just as short as software development is today -- seconds or minutes, instead of weeks or months.

And the greatest effect of this will be the freedom to think in terms of highly-concurrent hardware structures which implement the desired computation literally, in many cases directly, rather than by emulation or simulation. Von Neumann architectures will be used only when appropriate, while more organic customized architectures can begin to flourish.

The computer will *be* the computation desired at that moment.

Reconfigurable computing is not just a better way to do conventional designs, but a doorway to whole new domains.

Richard Shoup
Interval Research
Palo Alto, California
June 1994, revised August 1999

First published as the Foreword to Field-Programmable Gate Arrays, John V. Oldfield and Richard C. Dorf, Wiley-Interscience, 1994