The Last Byte

FPGAs, Programming Models, and Kit Cars

Vaughn Betz
University of Toronto

FPGA HARDWARE CAPABILITIES have grown tremendously in 30 years' time. In the early 1980s, an FPGA could implement only a few hundred gates of logic. Today's 28-nm FPGAs provide over 4,000 multiply-accumulate units, 10 million logic gates, thousands of on-chip RAM blocks, and myriad special-purpose units like PCI-express interfaces.

Their extreme degree of parallelism and ability to customize the RAM and data path architecture to the computation have enabled FPGAs to replace custom (ASIC) chips in many designs. Indeed, FPGAs are at the heart of a host of systems today, including very high throughput applications like wireless base stations and core packet routers. These systems can no longer justify the extremely high cost of ASIC development, but can justify the cost of a team of highly skilled engineers crafting HDL code. These are the Formula-1 teams of the FPGA world, optimizing data paths from the thousands of programmable elements in an FPGA.

FPGAs are useful for much more than ASIC replacements, however; diverse applications have shown performance improvements of an order of magnitude or more versus CPU- or DSP-processor–based implementations. The gains in power-efficiency are even more impressive, as FPGAs don't waste power fetching and interpreting instructions—they simply execute the computation. In this area, programming an FPGA in HDL is a hindrance, as most programmers aren't hardware designers. To them, using HDL to describe a custom engine to execute a computation, then writing the support code and instantiating the right IP interfaces to connect to the rest of the system doesn't feel like being part of a Formula-1 team—it's more like building a kit car when you just want to go for a drive.

But exciting developments on the horizon may change this situation. First, recent results in high-level synthesis (HLS) show that we can design many custom data paths in variants of C and achieve good performance while greatly improving productivity. For example, BDTI recently found that the AutoPilot HLS tool could implement signal processing applications in an FPGA with good productivity, and with 30× better cost-performance than a conventional DSP processor (www.bdti.com/Resources/BenchmarkResults/HLSTCP/AutoPilot). Second, more programmers are becoming parallel developers, as single-core CPU performance has stalled. This produces more developers who "think parallel" and also inspired languages like OpenCL that express parallelism in an augmented C variant. Parallel code should provide a more natural input to HLS than serial C, as the programmer has explicitly specified the parallelism, and parallelism is what hardware needs to gain performance. Moreover, OpenCL standardizes an abstraction between the "accelerator" and the CPU system. This eliminates the tedious task of writing low-level hardware and software to interface an FPGA into a CPU system—the compiler will now generate it automatically.

FPGA VENDORS ARE paying attention: Xilinx recently acquired AutoESL and Altera is investing heavily in an OpenCL compiler for FPGAs. Soon, accelerating a computation with an FPGA should feel a lot more like specifying options to build into your Ferrari, and less like building a car from a kit! ■

Vaughn Betz, an associate professor at the University of Toronto, was formerly senior director of software engineering at Altera. Contact him at vaughn@eecg.utoronto.ca.

■ Direct questions and comments about this department to Scott Davidson, Oracle, M/S USCA16-107, 4160 Network Circle, Santa Clara, CA 95054; scott.davidson@oracle.com.

0740-7475/11/$26.00 © 2011 IEEE Copublished by the IEEE CS and the IEEE CASS