ABSTRACT
In recent years, Field-Programmable Gate Arrays have become extremely powerful computational platforms that can efficiently solve many complex problems. The most modern FPGAs comprise effectively millions of programmable elements, signal processing elements and high-speed interfaces, all of which are necessary to deliver a complete solution. The power of FPGAs is unlocked via low-level programming languages such as VHDL and Verilog, which allow designers to explicitly specify the behavior of each programmable element. While these languages provide a means to create highly efficient logic circuits, they are akin to “assembly language” programming for modern processors. This is a serious limiting factor for both productivity and the adoption of FPGAs on a wider scale.

In this talk, we use the OpenCL language to explore techniques that allow us to program FPGAs at a level of abstraction closer to traditional software-centric approaches. OpenCL is an industry standard parallel language based on ‘C’ that offers numerous advantages that enable designers to take full advantage of the capabilities offered by FPGAs, while providing a high-level design entry language that is familiar to a wide range of programmers.

To demonstrate the advantages a high-level programming language can offer, we demonstrate how to use Altera’s OpenCL Compiler on a set of case studies. The first application is single-precision general-element matrix multiplication (SGEMM). It is an example of a highly-parallel algorithm for which an efficient circuit structures are well known. We show how this application can be implemented in OpenCL and how the high-level description can be optimized to generate the most efficient circuit in hardware. The second application is a Fast Fourier Transform (FFT), which is a classical FPGA benchmark that is known to have a good implementation on FPGAs. We show how we can implement the FFT algorithm, while exploring the many different possible architectural choices that lead to an optimized implementation for a given FPGA. Finally, we discuss a Monte-Carlo Black-Scholes simulation, which demonstrates the computational power of FPGAs. We describe how a random number generator in conjunction with computationally intensive operations can be harnessed on an FPGA to generate a high-speed benchmark, which also consumes far less power than the same benchmark running on a comparable GPU. We conclude the tutorial with a set of live demonstrations.

Through this tutorial we show the benefits high-level languages offer for system-level design and productivity. In particular, Altera’s OpenCL compiler is shown to enable high-performance application design that fully utilizes capabilities of modern FPGAs.

Categories and Subject Descriptors
B.6.1 [Logic Design]: Design Styles – Parallel Circuits

Keywords
FPGAs, OpenCL