Compiling Code Accelerators for FPGAs

Walid A. Najjar
University of California Riverside, Computer Science & Engineering
Riverside, California
+1.951.827.4406 najjar@cs.ucr.edu

ABSTRACT

This tutorial addresses the challenges and opportunities presented by compiled FPGA-based code accelerators.

In recent years we have witnessed a fast growth of both size and speed of FPGAs. These had been initially designed and marketed as convenient devices for "glue logic." Later, they became used as fast prototyping platforms. As their size and speed grew, they have been used for the short time to market they can afford. Lately, their size and speed have made them attractive as code accelerator. While the clock speed achievable on a typical FPGA design is about an order of magnitude lower than that on a typical CPU, their advantage comes from two sources: (1) Large degree of instruction and loop level parallelism. Parallel loops can typically be unrolled by factors ranging in the 100s. (2) Increased efficiency of hardware execution. The streaming of the data through a dedicated circuit eliminates a large number of support operations such as data fetch, address calculations, index management, loop control, etc. The combined higher efficiency and parallelism of hardware execution on FPGAs has been shown to result in speedups ranging from the 10s to the 1,000s over traditional processor on frequently executed code segments.

However, the main obstacle to wider acceptance of this technology is *programmability*. FPGAs are typically programmed using Hardware Description Languages (HDLs), which poses two problems: Traditional application developers are typically not HDL designers, and HDLs are not well suited for algorithm implementation. Furthermore, the FPGA is an amorphous mass of logic on which the compiler must create a data-path and schedule the computation. Such a task requires the harnessing of technologies developed for parallelizing compilers as well as those developed for high-level synthesis.

The main challenge that faces HLL to HDL translation is the paradigm shift from the stored program model to a value-based, data-driven execution – that is, from temporal to spatial execution. The task of an FPGA compiler is to generate both the data path and the sequence of operations (control flow) on that data path. The lack of architectural structure on the FPGA presents a number of opportunities for the compiler: (1) The available parallelism, instruction loop and thread, is very high and limited only by the size of the FPGA or the I/O bandwidth to the chip. (2) On-chip storage can be configured at will. (3) Circuit customization allows the compiler to reduce the circuit size as well as the clock duration.

Optimizing compilers for traditional processors have benefited from several decades of extensive research that has led to extremely powerful tools. Similarly, electronic design automation (EDA) tools have also benefited from several decades of research and development, leading to powerful tools that can translate VHDL and Verilog code, and recently SystemC code, into efficient circuits. However, little work has been done to combine these two approaches.

The Riverside Optimizing Compiler for Configurable Computing (ROCCC) is a C to VHDL compiler that targets the automatic generation of FPGA-based accelerators. ROCCC optimizes and parallelizes the most frequently executed loops for mapping as circuits on the FPGA. A host processor then manages the streaming of data through that circuit. The overall aim of ROCCC is to (1) bridge the performance gap between compiled and hand-written code and (2) apply extensive compile-time transformations on multi-dimensional arrays and non-trivial loop nests. Such transformations would be too complex for a human programmer to handle in a reasonable time. The objectives of the ROCCC optimizations are: (1) maximize the parallelism in the circuit as well as the clock rate at which it operates (2) minimize the number of off-chip memory accesses as well as the area of the circuit.

This tutorial will address the issues of compiling a high-level language to generate FPGA-based code accelerators. It will take a look at the whole field with a special emphasis on the ROCCC compiler toolset.

Tutorial outline:

1. FPGA code acceleration – An opportunity
2. Platform models - Why they matter
3. Compiling to FPGAs – The challenges
4. The ROCCC approach
5. The ROCCC toolset
6. Future outlook – Hardware, software and system support

Categories and Subject Descriptors

C.1.3 Computer Systems Organization, PROCESSOR ARCHITECTURES, Other Architecture Styles, Adaptable architectures
B.5.1 Hardware, REGISTER-TRANSFER-LEVEL IMPLEMENTATION, Design, Styles (e.g., parallel, pipeline, special-purpose).

General Terms

Design.

CODES+ISSS'07, September 30–October 5, 2007, Salzburg, Austria.
ACM 978-1-59593-824-4/07/0009.