Fast Fourier Transform on FPGA: Design Choices and Evaluation

Peter A. Milder, Franz Franchetti, James C. Hoe, and Markus Püschel

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA USA
{pam, franzf, jhoe, pueschel}@ece.cmu.edu

The discrete Fourier transform (DFT) is arguably the most important building block in digital signal processing applications. Fast algorithms for computing the DFT, called fast Fourier transforms (FFTs), exhibit concurrency and regularity that make them well suited for hardware implementation. Many different FFT implementations have been created for Field Programmable Gate Array (FPGA) platforms since the advent of this technology.

These implementations occupy a design space spanned by rather well-known degrees of freedom at different levels of design abstractions. However, an attempt to systematically assess the impact of the different design choices is very difficult because the available implementations in the literature are usually optimized for specific (and from case to case different) design specifications (such as signal length, precision, performance, or cost). Further, they target different generations of FPGA architectures.

In the Spiral project, we have developed a tool to automatically generate a large variety of RTL-synthesized FFT designs. An important property common to the FFT designs we produce is that they are efficient, that is all instantiated arithmetic elements are utilized in every cycle in the steady state. The tool supports design choices at different levels of abstraction, namely: different algorithms, different datapath architectures, and choices in the FPGA-specific mapping. This technology provides the opportunity to systematically evaluate the different FFT design points under a common tool flow and FPGA target, in this case Xilinx ISE 8.1i and Virtex-II Pro. In doing so, we can quantify the impact of the different choices available and provide guidelines for decision making in future FFT implementations.

We focus on two families of FFT algorithms most pertinent to hardware implementations. Additionally, we focus on two primary options in datapath architecture—fully streamed and horizontal-reuse datapaths, which are ideal for throughput and latency optimizations, respectively. We also explore low-level FPGA-specific considerations including the utilization of dedicated arithmetic units, efficient storage of twiddle constants, and methods for implementing data permutations. We consider the interdependencies between the decisions at these different levels of design abstractions. Lastly, we offer a systematic quantitative evaluation of RTL-level FFT implementations that are placed and routed for Xilinx Virtex-II Pro FPGAs. We observe that the combined degrees of freedom offered by these decisions result in a richly varied space of FPGA FFT implementations. Most notably, a wide range of tradeoffs between performance and resource requirements is available to suit application-specific requirements.