Using FPGAs and High Level Languages to Exploit the Inherent Parallelism of Multi-dimensional FFTs

Richard Chamberlain

Abstract—This paper examines using high level languages to program CPU/FPGA heterogeneous systems to perform floating point multi-dimensional FFTs. It investigates the degree of code redesign required versus performance increase and looks at the level of hardware knowledge required to achieve performance improvements. FFTs have been chosen as their inherent parallelism allows potential performance improvement when correctly distributed over parallel systems.

Index Terms— Field-programmable gate arrays, FFTs, High level languages, C

I. INTRODUCTION

Using FPGAs to accelerate FFTs is common place and can yield significant performance increases over CPU implementations. FFTs for FPGAs tend to be delivered as pre-optimized RTL (Register Transfer Level) logic libraries to be included as part of a larger RTL designs. This paper examines the performance achieved when writing an FFT using high level languages versus an optimized RTL version. It takes simple non optimized C code and compiles this to the FPGA using the Nallatech’s NAL (Nallatech Application Layer) design flow. The performance is progressively improved via small incremental design changes until the optimal performance is achieved.

The NAL design flow provides a C++ framework for describing CPU and FPGA heterogeneous systems, including all CPU to FPGA communications. User and library components can be connected together via simple communication channels. User components are described using a subset of C that can be compiled using the DIME-C compiler to create synthesizable RTL. DIME-C is a C to VHDL compiler provided as part for the NAL design tools.

II. BACKGROUND

The combination of large FPGAs and PCIe-gen2 FPGA accelerator cards increases the number of computational problems suitable for acceleration using CPU and FPGA heterogeneous systems.

Large multi-dimensional FFT’s suffer significant performance degradation on CPUs due to the limited size of an individual cores cache. A unique caching structure can be created by combining FPGA logic with external memories to provide a system not bound by the same limitations.

III. OVERVIEW

Using the NAL tool flow we were able to show that is possible to quickly design and accelerate complex algorithms on FPGAs in a time comparable to programmable a CPU. Here all FFT code including the FFT setup and calculation of scaling factors are calculated on the FPGA.

To run the FPGA accelerated FFT the host CPU sends the size and direction of the FFT to the FPGA via a PCIe-gen2 interface. The FPGA then creates a lookup table for the necessary scaling coefficients. The host then sends the complex floating point data to be transformed as either a 1D or 2D data set.

When programming the FFT the NAL software tool flow provides information regarding the state of the key processing loops. To achieve maximum performance it is imperative that the all performance sensitive loops are pipelined or exhibit low latency. To further improve performance the kernel code responsible for performing the FFT butterfly calculations is replicated multiple times using a multiplexer to partition data between the multiple kernels. In all 12 kernels were required to achieve maximum performance.

As NAL code is standard C++ it can be compiled and executed on a standard x86 CPU, providing a test and debug timescale similar to CPU or GPGPU programming environment. The NAL design includes all host and interface control logic by default with a simple compiler option required to change the software model into a FPGA accelerated library.

Using a Nallatech PCIe280 accelerator card populated with an LX330 Virtex5 FPGA, the design was capable of performing a 1024x1024 FFT in approximately 5.0 milliseconds. This was benchmarked against an Intel 1.88GHz Xeon quad core processor achieving 7.5x improvement over a single core or 3x improvement over all 4 cores.

1 Assumes the user is experienced in parallel programming and has some knowledge of FPGA resources and capability.

This work was sponsored by Nallatech Ltd.

Richard Chamberlain is with Nallatech Ltd., Boolean House, 1 Napier Park, Glasgow, UK, G68 0BH (e-mail: r.chamberlain@nallatech.com).