CS97: Project Proposal
Determining the Offloading Point for Image Processing: A Software versus Hardware Comparison

Danielle Sullivan, Eliza Bailey, Taylor Nation
Swarthmore College

October 4, 2013

Abstract

For this project we have decided to determine at what threshold point it is quantifiable, according to our metrics, more effective to run image processing on hardware (FPGA) as opposed to software (C++, python code). We will determine our own set of metrics used to measure the performance of the image processing. Our experiments will include trying two different types of images manipulation, possibly FFT and Convolution, and programming two different hardwares (2 different FPGAs). The performance results generated will be compared to results generated by processing images on a computer using C++ or python code. Our goal is to determine exactly at what point, and under what conditions, it is advisable to offload from software to hardware.

Motivation

Digital image processing is a rapidly growing field with an extremely wide range of applications. As the sizes and bit depth of images grow, software implementation of DIPs grows increasingly less applicable. This comes increasingly true for real-time applications. Recently research has shown that offloading these techniques onto hardware has significant speed up. Therefore finding the offloading point between running DIPs on software versus hardware is important.

PFGAs offer many advantageous qualities such as high performance, optimization, computational density, and reliability. In addition PFGAs are know for being highly reliable and having low costs. Finally FPGAs are truly parallel, making them very attractive for computationally expensive tasks which could easily slow down a computer [1].

Background

FPGAs' utility lies within their ability to take some of the load off of CPU/GPUs, and perfom multi-threaded applications. Much in the same way that transferring data to the GPU for computation incurs some overhead, transferring call to the FPGA requires a performance slowdown. Recent research involving the interplay between the FPGA and the CPU include papers on designing a way to offload certain operations from the CPU (which generally is accessed via software) to the FPGA (which requires knowledge of hardware design). Patyk, Salmela, et al [2], in their 2011 paper "Design Methodology for Offloading Software Executions to FPGA", designed a method that converts C code to a format compatible with FPGAs.

In Patyk's experiment, the researchers used a code-design tools to translate from a higher level language into a FPGA-readable instruction set that uses a Transfer-Tered Architecture (TTA). The C-code is replaced with calls to the device. The developer then creates a definition of the processor/hardware interaction, and using this definition, generates machine code for the FPGA. This mapping is not platform independent; new code is generated for different architecture. The resulting experiments using the FPGA offloading were successful, drastically slowing down the amount of clock cycles it took to complete the computation.

Our Idea

Through this project we are hoping to investigate the threshold point, offloading point, between deciding to process an image using software on a computer or using the hardware on an FPGA. We believe there is a certain point, based on program size, for which it is more advantageous to run an image process using hardware. Our project will consist of us trying to determine under what conditions it is preferable to use hardware. To accomplish this we will use time and possibly power consumption as metrics for determining our performance level. We will experiment with two different types of image processes (possibly FFT and convolution) and two different processors (2 Altera FPGA models). We will also run our image manipulations on a regular computer using simple (C++ or python) code as a way to compare our results.

Milestone Goals

This section describes the milestones we have set for ourselves.

Milestone 1

Our first milestone will be to run image processing code we get from an outside source on our FPGA. This milestone ensure we learn how to use our FPGA, and have the fundamentals necessary to achieve future milestones. At this time, we will have also researched multiple options for image processing techniques we could potentially implement. Initially we believe we will explore processes using FFT and convolution, but this may change with further research.

Milestone 2

Our second mile stone will include researching off loading metrics to determine if running certain image processing techniques is more effectively done through hardware (FPGA) or through software implementation(C++ or python code). Our initial thoughts include metrics based on time and power consumption. Our research will help us determine how we will do this, and if there are other metrics worth exploring. Additionally, by this mile stone we should know which 2 image processing techniques we will analyzing in our research,and have their software and hardware implementations running.

Milestone 3

Our third milestone we will have determined the metric we will be using, and have it's implementations running. If we have been able to reach all of our milestones thus far, we will preform the same analysis on a different Altera FPGA model.

Citations

[1] http://home.engineering.iastate.edu/ sparsh/FPGA _ ImageProcessing.pdf
[2] http://link.springer.com/article/10.1007% 2Fs11265-011-0606-x#page-4