2D Phase Unwrapping on FPGAs and GPUs

Sherman Braganza, Miriam Leeser
Northeastern University
Boston, MA
{sbraganz, mel}@coe.neu.edu

In this paper we present two implementations of the minimum L^P norm phase unwrapping algorithm. This computation involves a 2D Discrete Cosine Transform (DCT) of 1024x512 and represents amongst the largest DCT/IDCTs on an FPGA documented in the literature. The other platform is an Nvidia 8800GTX Graphics Processing Unit (GPU) on which the algorithm is implementing using their Compute Unified Device Architecture (CUDA) API.

Introduction

There exist several applications that make use of coherent signals for imaging purposes. Coherent signals contain information about both magnitude and phase as opposed to incoherent ones that just contain magnitude information. Applications utilizing such signals include Synthetic Aperture Radar (SAR), Magnetic Resonance Imaging (MRI), optical interferometry and adaptive beamforming. Such applications often have a reference signal to which the received signal is compared (a stable local oscillator located in the radar unit in the case of SAR) and from that comparison the phase is extracted. However, this extraction is limited by the fact that the output phase will lie between π and $-\pi$. Hence, the raw output phase is referred to as *wrapped*.

Given a noise free signal, the original phase can be recovered by accumulating the phase difference and an integer multiple of 2π every time a discontinuity is detected. In the presence of noise however, such an unwrapping can fail catastrophically in the two dimensional case.

Phase Unwrapping

A set of methods has evolved as a solution to this problem, discussed and analyzed in [1]. They can be roughly classified into two groups: those that unwrap around noisy sections and those that use numerical error minimization techniques. One of the methods that consistently produces high quality results is the minimum L^P norm technique. This iterative technique, which falls under the class of error minimization algorithms, operates by first making an initial guess as to the solution and then iteratively refining it. It uses the minimum P (i.e. P=0) since that allows the final data to match the measured data in as many places as possible (if P were set to 2, this would be the equivalent of a least squares minimization). Convergence of the algorithm is tested by checking for residues. If convergence is not achieved, the algorithm terminates after a preset number of iterations. A residue occurs when a closed loop integral on the two dimensional image data does not produce a zero, but instead produces either 2π or -2π. The presence of residues indicates that naïve unwrapping via accumulation cannot be performed.

A sample of the results produced by the minimum L^P norm phase unwrap is displayed below in Figure 1.

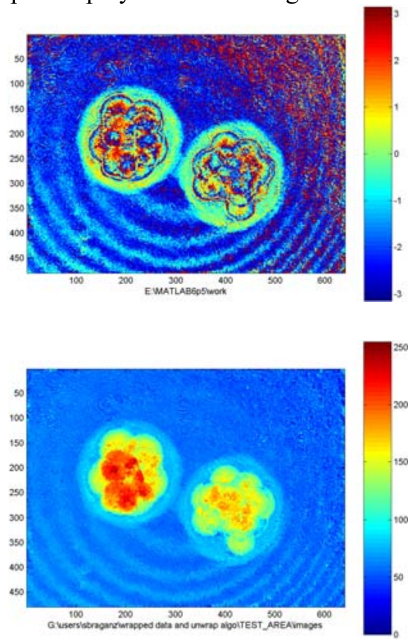
E:\MATLAB6p5\work
G:\users\sbraganz\wrapped data and unwrap algo\TEST_AREA\images

Figure 1: Wrapped phase above and unwrapped phase below. The image is of a mouse embryo and was taken using Optical Quadrature Microscopy (OQM), an interference based microscopy method.

The phase unwrap algorithm utilizes a 2D DCT, a Poisson calculation and then a 2D IDCT. The DCTs and Poisson calculation consume 70% of the total processing time and thus are a prime target for speedup.

Implementation

The FPGA implementation exploits the separability property of the DCT to decompose the 2D input matrix into 1D rows, compute the transform, transpose the matrix and then recompute. This theoretically allows multiple 1D cores (described in [1]) to run in parallel. Intermediate computation of the frequency domain Poisson equation is then performed in a streaming fashion followed by the 2D IDCT. The platform used was the Annapolis Wildstar II Pro, a Virtex II Pro based accelerator board.

The initial GPU implementation implements the same kernel (DCT, Poisson, IDCT) on the 8800GTX but performs entire 2D DCTs using identically sized 2D FFTs[2]. The FFT is done with a single kernel on the GPU; on the FPGA it was decomposed into rows and columns.

Since program space on the GPU is less limited than on the FPGA, a larger portion of the phase unwrap algorithm was subsequently implemented. The portion implemented on the GPU included several matrix operations including constant bias elimination and Laplace computation in addition to the original kernel. To implement phase unwrapping, these computations iterate a large number times. Implementation of this larger segment of code on the GPU results in a drastic reduction in data transfer overhead since data remains on the GPU board.

Results

The DCT and Poisson calculations consume approximately 70% of the total processing time and thus the upper bound on speed-up is 3x, assuming zero data transfer and computation time for the DCT. The entire inner kernel however consumes around 90% and thus has an upper bound of 10x.

A working demonstration on the Wildstar II Pro with a single 1D DCT core operates at 100 MHz and consumes 37% of the total area. It completes the transforms and Poisson calculation in 82 ms including data transfer times. However, the Virtex II Pro is a six year old platform. For a fair comparison, synthesis results from a Virtex 5 show that it can operate at frequencies of 190 MHz in addition to which at least one more DCT core can be integrated into the design. Assuming a data transfer time similar to that of a GPU, that results in a 19 ms operation for two cores and 14ms for three.

The Nvidia implementation takes 13.9 ms assuming that the same logic is implemented on the GPU as on the FPGA. This performance is relatively similar to the FPGA. However, once the larger program segment is implemented on the GPU, overall algorithm performance more than doubles.

Conclusions

FPGAs are superior as front-end processors, processing data straight from sensors. When the data to be processed is already on a host computer, the choice is less clear. An architecture that minimizes communications over slow interconnect such as PCI has a definite advantage, as does an implementation that minimizes the amount of data to be communicated. For this example, the GPU has superior host to chip communications, as well as the ability to do more processing on the chip, minimizing data transfer time. The nature of the phase unwrap algorithm is such that it consists of many sequential and dependent matrix operations. Reprogramming an FPGA with a new kernel would consume too much time to be efficient, so the GPU with it's larger program space provides the better solution. In addition, the GPU was easier to program. For phase unwrap, the GPU implementation was superior in overall application acceleration.

Acknowledgements

This work was supported in part by CenSSIS, the Center for Subsurface Sensing and Imaging Systems, under the Engineering Research Centers Program of the National Science Foundation (award number EEC-9986821).

References

[1] D. C. Ghiglia and M. D. Pritt. *Two-Dimensional Phase Unwrapping: Theory, Algorithms and Software*. Wiley Inter-Science, 605 Third Avenue, New York, NY, 10158-0012,1998.

[2] J. Makhoul. "A fast cosine transform in one and two dimensions". *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):27–34, February 1980.

[3] Xilinx Inc. Fast Fourier Transform 3.2. http://www.xilinx.com/ipcenter/catalog/logicore/docs/xfft.pdf, Last accessed March 2007.

[4] Sherman Braganza and Miriam Leeser. "Phase unwrapping on Reconfigurable Hardware", *High Performance Embedded Computing*, 2007.