Hyperspectral Image Compression on Reconfigurable Platforms[1]

Thomas W. Fry
Department of Electrical Engineering
University of Washington
Seattle, WA 98195
tom@tomfry.com

Scott Hauck
Department of Electrical Engineering
University of Washington
Seattle, WA 98195
hauck@ee.washington.edu

ABSTRACT

NASA's satellites currently do not make use of advanced image compression techniques during data transmission to earth because of limitations in the available platforms. With the advent of Field Programmable Gate Arrays (FPGAs) and Adaptive Computing technologies it is now possible to construct a system, which compresses the data stream before down linking. Our work is part of a NASA-sponsored study on the design and implementation of FPGA-based Hyperspectral Image Compression algorithms for use in space.

In this paper we present an implementation of the SPIHT image compression routine in reconfigurable logic. SPIHT is a progressive wavelet-based image compression coder. It first converts the image into its wavelet transform and then transmits information about the wavelet coefficients. We discuss both memory storage considerations and optimizations to the original SPIHT algorithm for use in hardware. To fully utilize all of the memory bits for each wavelet coefficient and reduce memory usage, we introduce the concept of *Variable Fixed-Point* representation.

The paper also presents a modification to the original SPIHT algorithm needed to parallelize the computation. The architecture of the SPIHT engine is based upon Fixed-Order SPIHT, developed specifically for use within adaptive hardware. For an N x N image, Fixed-Order SPIHT may be calculated in $N^2/4$ cycles. Square images which are powers of 2 up to 1024 x 1024 are supported by the architecture we implemented. Our system was developed and tested on an Annapolis Microsystems WildStar board populated with Xilinx Virtex-E parts.

1. Introduction

As NASA deploys satellites with more sensors, capturing an ever-larger number of spectral bands, the volume of data being collected is beginning to outstrip satellite's transmition channels. At the same time the volume of data being collected is growing at a faster rate than improvements in transmition capabilities, making methods of compressing images prior to down linking necessary.

Current technologies are unable to provide NASA with a viable platform to process data in space. Software solutions suffer from performance limitations and power requirements. At the same time traditional hardware platforms lack the required flexibility needed for post-launch modifications. By implementing an image compression kernel in a reconfigurable system, it is possible to overcome these shortcomings. Since such a system may be reprogrammed after launch, it does not suffer from conventional hardware's inherit inflexibility. Yet the algorithm is computing in custom hardware and can perform at the required rates, while consuming less power than a traditional software implementation.

Our work is part of a NASA-sponsored investigation into the design and implementation of a space-bound FPGA-based Hyperspectral Image Compression algorithm. We have selected the Set Partitioning in Hierarchical Trees (SPIHT) compression routine and optimized the algorithm for implementation in hardware. This thesis describes our work towards this effort and provides a description of our results.

2 Description of the Algorithm

SPIHT is a wavelet-based image compression coder. It first converts the image into its wavelet transform and then transmits information about the wavelet coefficients. The decoder uses the received signal to reconstruct the wavelet and performs an inverse transform to recover the image. We selected SPIHT because it displays exceptional characteristics over several properties all at once [4]. They include:

- Good image quality with a high PSNR
- Fast coding and decoding
- A fully progressive bit-stream
- Can be used for lossless compression
- May be combined with error protection
- Ability to code for exact bit rate or PSNR

[1] This paper is an abstract of the full paper: T. Fry, S. Hauck "Hyperspectral Image Compression on Reconfigurable Platforms", *IEEE Symposium on Field-Programmable Custom Computing Machines,* 2002.

SPIHT is a method of coding and decoding the wavelet transform of an image. By coding and transmitting information about the wavelet coefficients, it is possible for a decoder to perform an inverse transformation on the wavelet and reconstruct the original image. The entire wavelet does not need to be transmitted in order to recover the image. Instead, as the decoder receives more information about the wavelet, the inverse-transformation will yield a better quality reconstruction of the original image [3].

SPIHT codes a wavelet by transmitting information about the significance of a pixel compared to some threshold, thus implying some information about the pixel's value. To take advantage of redundancies within a wavelet, SPIHT transmits information stating whether a pixel or any of its descendants are above a threshold. At the end of each pass the threshold is divided by two and the algorithm continues. By proceeding in this manner, information about the most significant bits of the wavelet coefficients will always precede information on lower order significant bits, which is referred to as bit plane ordering.

In addition to transmitting wavelet coefficients in a bit plane ordering, the SPIHT algorithm develops an individual order to transmit information within each bit plane. The ordering is implicitly created from the threshold information discussed above and by a set of rules which both the encoder and decoder agree upon. Thus each image will transmit wavelet coefficients in a variable order dependent upon the image's wavelet transform. Slightly better Peak Signal to Noise Ratios (PSNR) are achieved by using this dynamic ordering of the wavelet coefficients. The trade-off for the improvement are increased run-times for both the encoder and decoder since the order must be considered.

3 Design Considerations and Modifications

In order to fully take advantage of the high performance a custom hardware implementation of SPIHT can yield, the software specifications must be examined and adjusted where they either perform poorly in hardware or do not make the most of the resources available. Here we discuss both memory storage considerations and optimizations to the original SPIHT algorithm for use in hardware.

3.1 Variable Fixed-Point

The discrete wavelet transform produces real numbers as wavelet coefficients. Traditionally FPGAs do not employ the use of floating-point numbers because of their lower performance and consumption of hardware resources. Since coefficients at each wavelet level of the DWT have a fixed numerical range, we opted for a fixed-point numerical representation.

One property of the DWT is the numerical range of numbers possible within each wavelet level is fixed, yet varies between levels due to the 2-D low-pass FIR filter. As a result, coefficients at various wavelet levels require a variable number of bits above the decimal point to cover their possible ranges. Another property of the DWT is the number of bits used to represent each coefficient impacts the Peak Signal-to-Noise Ratio (PSNR) of the resulting image. Figure 1 shows the average PSNR of several images coded with a variable number of bits. An assignment of 16 bits per coefficient most accurately matches the full precision floating-point coefficients used in software and was selected.

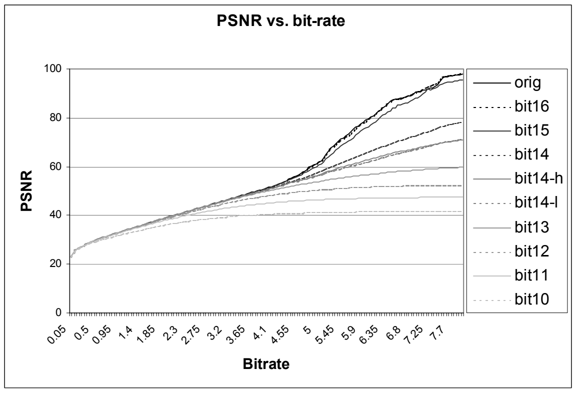

Figure 1: PSNR vs. bit-rate for various coefficient sizes

To fully utilize all the bits for each wavelet coefficient, we introduce the concept of *Variable Fixed-Point* representation. With Variable Fixed-Point we assign a fixed-point numerical representation for each wavelet level optimized for the expected data. In addition, each representation differs from one another, meaning we employ a different fixed-point scheme for each wavelet level. Doing so allows us to optimize both memory storage and I/O at each wavelet level to yield maximum performance.

3.2 Fixed Order SPIHT

As discussed within Section 2 the SPIHT algorithm computes a dynamic ordering of the wavelet coefficients as it progresses. Such an ordering will yield better image quality for bit-streams which end within the middle of a bit-plane. The drawback of this ordering is that every image will have a unique list order determined by the image's wavelet coefficient values.

Yet, the data that a block of coefficients contributes to the final SPIHT bit-stream is fully determined by a set localized information. Thus, every block of coefficients may be calculated independently and in parallel of one another. However, the order that a block's data is inserted into the bit-stream is not known since this order is dependent upon the image's unique ordering. However the algorithm employed to calculate the ordering of coefficients is sequential in nature and can not be parallelized in hardware, significantly limiting the throughput of any implementation.

We propose a modification to the original SPIHT algorithm called *Fixed Order SPIHT*. In Fixed Order SPIHT the order in which blocks of coefficients are transmitted is fixed before hand. Doing so removes the need to calculate the ordering of coefficients within each bit-plane and allows us to create a fully parallel version of the original SPIHT algorithm. Such a modification increases the through put of a hardware encoder by greater than an order of magnitude, at the cost of a slightly lower PSNR within each bit-plane (approximately 0.1 – 0.2 dB). For a more complete discussion on Fixed Order SPIHT refer to Fry et al. [2].

4 Architecture

4.1 Target Platform

Our target platform is the WildStar FPGA processor board developed by Annapolis Micro Systems [1]. The board consists of three Xilinx Virtex 2000E FPGAs. It operates at rates up to 133MHz with 48MBytes of memory available through 12 independent ports.

4.2 Design Overview

Our architecture consists of three phases: Wavelet Transformation, Maximum Magnitude Calculation and Fixed Order SPIHT Coding. Each phase is implemented in one of the three Virtex chips. By instantiating each phase on a separate chip, separate images can be operated upon in parallel. Data is transferred from one phase to the next through the shared memories. By coding a different image in each phase simultaneously, the throughput of the system is determined by the slowest phase, while the latency of the architecture is the sum of the three phases. Figure 2 illustrates the architecture of the system.

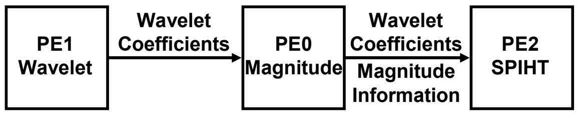

Figure 2: Overview of the architecture

5.3 DWT Phase

For the DWT phase we designed a folded architecture which processes one dimension of a single wavelet level. Pixels are read in horizontally from one memory port and written directly to a second memory port. In addition pixels are written to memory in columns, inverting the image along the 45-degree line. By utilizing the same addressing logic, pixels are again read in horizontally and written vertically. However, since the image was inverted along its diagonal, the second pass will calculate the vertical dimension of the wavelet and restore the image to its original orientation. Each dimension of the image is reduced by half and the process iteratively continues for each wavelet level. To speed up the DWT, the design reads and writes four rows at a time. Figure 3 illustrates the architecture of the discrete wavelet transform phase.

Since every pixel is read and written once and the design processes four rows at a time, for an N x N size image both dimensions in the lowest wavelet level will compute in N/4 clock cycles. Similarly, the next wavelet level will process the image in ¼ the number of clock cycles as the previous level. With an infinite number of wavelet levels the image will process in:

$$\sum_{l=1}^{\infty} \frac{2 \cdot N^2}{4^l} = \frac{3}{4} \cdot N^2$$

Thus bounding the runtime of the DWT engine is bounded by ¾th a clock cycle per pixel in the image.

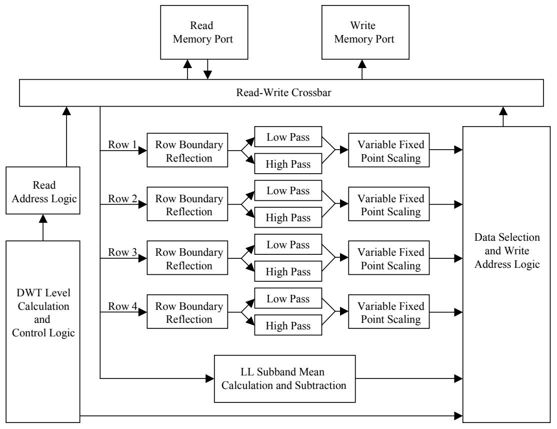
Read Memory Port
Write Memory Port
Read-Write Crossbar
Read Address Logic
DWT Level Calculation and Control Logic
Row 1
Row 2
Row 3
Row 4
Row Boundary Reflection
Low Pass
High Pass
Variable Fixed Point Scaling
Data Selection and Write Address Logic
LL Subband Mean Calculation and Subtraction

Figure 3: DWT Architecture

5.4 Maximum Magnitude Phase

The maximum magnitude phase calculates and rearranges the following information for the SPIHT phase. 1) The maximum magnitude of each of the 4 child trees 2) The maximum magnitude of the current tree 3) Threshold and Sign data of each of the 16 child coefficients and 4) Re-orders the wavelet coefficients into a Morton Scan ordering. It does do by implementing a stack and reads each pixel in a depth first search ordering so that child pixels are always read before parent pixels.

5.5 SPIHT Phase

The final SPIHT Coding phase essentially computes the parallelize algorithm. Coefficient blocks are read from the highest wavelet level to the lowest. As information is loaded from memory it is shifted from the Variable Fixed Point representation to a common fixed point representation for every wavelet level. Once each block has been adjusted to the same numerical representation, the parallel version of SPIHT is used to calculate what information each block will contribute to each bit plane.

The information is grouped and counted before being added to three separate variable FIFOs for each bit plane. The data which the variable FIFO components receive varies in size, ranging from zero bits to thirty-seven bits. The variable FIFOs are used to arrange the block data into regular sized 32-bit sized words for memory accesses. Care is also taken to stall the algorithm if anyone of the variable FIFOs becomes too full. The block diagram for the SPIHT coding phase is given in Figure 4.

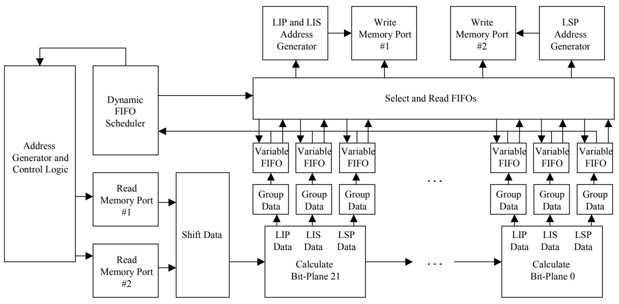

Figure 4: SPIHT Coding Phase Block Diagram

6 Design Results

Our system was designed using VHDL with models provided from Annapolis Micro Systems to access the PCI bus and memory ports. Simulations for debugging purposes were done with ModelSim EE 5.4e from Mentor Graphics. Synplify 6.2 from Synplicity was used to compile the VHDL code and generate a net list. The Xilinx Foundation Series 3.1i tool set was used to both place and route the design. Lastly the peutil.exe utility from Annapolis Micro Systems generated the FPGA configuration streams.

Table 1 shows the speed and runtime specifications of our architecture. All performance numbers are measured results from the actual hardware implementations. Each phase computes on separate memory blocks, which can operate at different clock rates. The design can process any square image where the dimensions are a power of 2: 16 by 16, 32 by 32 up to 1024 by 1024.

We compared our results to the original software version of SPIHT provided on the SPIHT website [4]. The comparison was made without arithmetic coding since our hardware implementation currently does not perform any arithmetic coding on the final bit-stream. An Ultra 5 SPARC workstation was used for the comparison and we used a combination of satellite images from NASA's website and standard image compression benchmark images. The software version of SPIHT compressed a 512 x 512 image in 1.14 seconds on average. The wavelet phase, which constrains the hardware implementation, computes in 2.48 milliseconds, yielding a speedup of **457** times for the SPIHT engine. In addition, by creating more parallelized implementation of the wavelet phase, further improvements to the runtimes of the SPIHT engine are possible.

7 Conclusions

In this paper we demonstrated a viable image compression routine on a reconfigurable platform. We showed how by analyzing the range of data processed by each section of the algorithm, it is advantageous to create optimized memory structures as with our Variable Fixed Point work. Doing so minimizes memory usage and yields the utmost usefulness of transferred data. (i.e. each bit transferred between memory and the processor board directly impacts the final result.) In addition our Fixed Order SPIHT work illustrates how by making slight adjustments to an existing algorithm, it is possible to dramatically increase the performance of a custom hardware implementation and simultaneously yield essentially identical results. With Fixed Order SPIHT the throughput of the system increases by more than two orders of magnitude while still matching the original algorithm's PSNR curve.

Table 1: Performance Numbers

Phase	Clock Cycles per 512x512 image	Clock Cycles per Pixel	Clock Rate	Throughput	FPGA Area
Wavelet	182465	3/4	75 MHz	100 MBytes/sec	62%
Magnitude	131132	1/2	73 MHz	146 MBytes/sec	34%
SPIHT	65793	1/4	56 MHz	224 MBytes/sec	98%

Our SPIHT work is part of an ongoing development effort funded by NASA. Future work will to address how lossy image compression will affect downstream processing. The level of lossy image compression that is tolerable before later processing begins to yield false results needs to be analyzed and dealt with. Lastly improvements to SPIHT and the consequences to a hardware implementation will be studied. Modifications to Fixed Order SPIHT including adding error protection to the bit-stream and region of interest coding will be considered.

8 References

[1] Annapolis Microsystems. *WildStar Reference Manual*, Maryland: Annapolis Microsystems, 2000.

[2] T. W. Fry, *Hyper Spectral Image Compression on Reconfigurable Platforms*, Master Thesis, University of Washington, Seattle, Washington, 2001.

[3] A. Said, W. A. Pearlman, “A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, pp 243 - 250, June 1996.

[4] A. Said, W. A. Pearlman, “SPIHT Image Compression: Properties of the Method”, http://www.cipr.rpi.edu/research/SPIHT/spiht1.htm l