PRECISION FOR 2-D DISCRETE WAVELET TRANSFORM PROCESSORS

Michael Weeks
Department of Computer Science
Georgia State University
Atlanta, GA 30303
E-mail: mweeks@cs.gsu.edu

Abstract: The 2-D Discrete Wavelet Transform (DWT) suits image processing applications well, allowing for excellent compression. Many architectures have been proposed to perform the 1, 2 and 3-D DWT, but few address the precision necessary to ensure perfect reconstruction. The goal of this work is to experimentally determine the precision needed to store the results of a 2-D DWT without introducing round-off error via the 2-D DWT processor.

1. INTRODUCTION

Wavelet analysis applies to many different applications, such as digital communications, biomedical signal processing, medical imaging, matrix computation, digital signal compression, and video-conferencing [1]. Wavelets can be used to represent data as diverse as heartbeats and television signals, in a way that reduces redundancy within the signal. Therefore, it can be compressed and stored in less space after the transform. One example application is the wavelet-based real-time compression system for video conferencing [2]. But using floating-point arithmetic limits the wavelet transform's potential, since the time needed is unsuitable for real-time applications. An alternative is to use fixed-point arithmetic, which this paper explores.

The most significant use for the DWT so far has been in compression, especially for video coding applications [3]. Many architectures have been proposed, such as [4] specifying a pipelined VLSI architecture for adaptive digital image compression. Compression has three steps. The first step is the transform, which represents the data in a different form, without losing information. The second step, quantization, maps data values to a finite set, which loses information. The name “lossy” compression comes from quantization. The third step encodes the data in a more compact way, such as with Huffman coding. For example, the sequence 11000000 would be encoded much like a person would say it: as "two ones followed by six zeroes". The sparser the transform output is, the more compact the encoding will be.

The Discrete Wavelet Transform (DWT) is a discrete-time and discrete-scale basis transformation. In other words, though the DWT produces real (i.e. floating-point) values, it indexes them with integer time and scale values. The focus of this work is to determine the amount of precision needed if fixed-point numbers are used to store these values instead of floating-point. In the DWT, a filter transforms a discrete-time signal into an average signal (from a low-pass filter) and a detail signal (from a high-pass filter). This specifies 1 octave, where an octave is a level of resolution. For the 2-

Dimensional case, a pair of filters operate on the data horizontally, then vertically, as shown in Figure 1. With multiresolution analysis, the average signal is piped into another set of filters, which produces the average and detail signals at the next octave [5]. The detail signals are kept, but the higher octave averages can be discarded, since they can be computed during the synthesis. Each octave's outputs have only half the input's amount of data, thus, the wavelet representation is the same size as the original.

The next section covers the DWT background in greater detail. The experiment follows in section 3, and the final section presents results and conclusions.

2. DWT BACKGROUND

A pair of waveforms generate wavelets: the wavelet function and the scaling function. As the names suggest, the wavelet function produces the wavelets, while the scaling function finds the approximate signal at that scale. The analysis procedure moves and stretches the waveforms to make wavelets at different shifts (starting times) and scales (durations). The resulting wavelets include coarse-scale ones that have a long duration, and fine-scale ones that last only a short amount of time.

Discrete wavelet transformers multiply the input signal by the shifts (translation in time) and scales (dilations or contractions) of the wavelet. J represents the total number of octaves (levels of resolution), while j is used as an index to the current octave (1 <= j <= J). N stands for the total number of inputs, and n indexes the input values (1 <= n <= N). $W_h(n,j)$ represents the DWT output (detail signals). W(n,0) indicates the input signal, and W(n,j) gives the approximate signal at octave j [6]. In the equations below, "h" means the coefficients for the low-pass filter, and "g" means the coefficients for the high-pass filter. See [7] for more information.

The low-pass output is:

$$W(n, j) = \sum_{m=0}^{2n} W(m, j-1) * h(2n - m)$$

and the high-pass output is:

$$W_h(n, j) = \sum_{m=0}^{2n} W(m, j-1) * g(2n - m)$$

The fast pyramid algorithm, by Stephanie Mallat and Yves Meyer [5], quickly computes the 1-D wavelet transform. The algorithm gets its efficiency by halving the output data at every stage, otherwise known as down sampling, giving it a complexity of O(N) [6]. Note that every octave divides the value n by 2, since the DWT outputs are downsampled at every octave. Since a DWT filter only keeps half of the filter outputs, then only half need to be computed [8]. The wavelet filters generate $N/2^j$ outputs for each octave, for a total of N/2+N/4+N/8+...+1 = N outputs. The scaling

filters also generate $N/2^j$ values, but these are only used internally (they are inputs to the next pair of filters), except for the last octave. The maximum number of octaves is based on the input length, $J = \log_2(N)$, however, practical applications limit the number of octaves, typically to J=3.

Figure 1 shows a 2-Dimensional DWT. The input signal goes to the low-pass and high-pass filters of the horizontal stage. The high-pass filter multiplies the input signal by a transposed vector of the wavelet function coefficients, and the low-pass filter multiplies the input signal by a transposed vector of the scaling function coefficients. The outputs of each filter are downsampled, meaning that every other value is discarded. At this point, the signals can be compressed, or transmitted, or processed as desired. In the synthesis stage - the inverse DWT - the signals are first upsampled. Upsampling stretches the data out, inserting 0's between each signal value. Next, the signals are multiplied again by vector transposes in parallel, and the results are added together. The output signal is exactly the same as the input signal, unless the signal undergoes a lossy compression in the wavelet domain.

2.1 Fixed Point Versus Floating Point

When designing a wavelet transform processor, one question is how to treat the results, as fixed-point values, or floating point? Fixed point has the advantages of being easier to implement, requires less silicon area, and makes multiplications faster to perform. Floating point allows a greater range of numbers, though floating point numbers require 32 or 64 bits. Fixed-point numbers can be 8 to 32 bits (or more), possibly saving space in the multiplier.

The wavelet transform has compact support, and the coefficients have small sizes that do not take advantage of the floating-point number's wide range [9]. Also, using fixed-point numbers allows for a simpler design. In other words, the extra hardware that floating point requires is wasted on the DWT. A standard design for integer multiplications is the high speed Booth multiplier, as used by [9]. The paper [10] echoes this choice by incorporating a high-speed Booth multiplier for DWT filtering in a video compression system.

Limqueco implemented his design with fixed-point computations [11]. In the horizontal and vertical filters, decimal numbers including some bits for the fractional part represent the values. When the results leave the filters, the values are rounded and truncated to an 8-bit format.

Reference [10] also uses fixed-point numbers, and the author notes that the DWT coefficient's range will grow for the 1-D case. The increase is upper-bounded by approximately 2. An extra bit of precision for each transform octave or dimension will be needed. For a 1-D architecture, Grzeszczak also notes that 12 bits of precision are enough [10]. Other designers also use fixed point, such as [12], who use 8 bit fixed-point arithmetic for data and coefficients. Results were rounded back to 8 bits for the next stage of computation. Wang and Huang write that wavelet transformed data is naturally floating point, but they round it to the nearest integer values, to minimize data loss [13]. Aware's chip has 16x16 size multipliers, but keep 16 bits of the result, which is chosen by software. It has 16 bit adders, and its I/O data size is 16 bits [14]. Therefore, fixed-point numbers with an assumed radix have been shown to work for computing the DWT.

2.2 Precision

Nine grey-scale images were used in this study, including Figure 2. These input values are 8 bits wide, and the arithmetic result for 1 filtering operation has 16 bits (assuming an 8 bit x 8 bit multiplication). The question is how much precision to keep? If all multiplication bits are kept every time, then the size of the coefficients will grow by a factor of 3 just for 1 octave of the 2-D DWT. Eight bits just is not enough space; packing the data into 8 bits loses some precision. When precision is lost, an exact reconstruction is not guaranteed. Even with Haar coefficients (1/2, 1/2, and 1/2, -1/2), there must be one extra bit for the sign, so the width of the intermediate data "grows" by 1 bit. If truncation were used instead, keeping the lower 7 bits and using the 8th bit as the sign, the reconstruction would not be able to distinguish a black pixel value "0" from a mid-grey "128". Most previous work that address this use either 8 or 16 bits in their 1-D designs, e.g. Aware's processor. One author (Grzeszczak [10]) does some analysis on the word size, and shows that 32 bits is too much for the 1-D case. This paper explores the 2-D case, experimentally.

3. PRECISION EXPERIMENT

To experimentally find how much precision is necessary, 2-D images were transformed, and the results were stored as fixed-point values. The transform uses the DWT package provided with Matlab, since only storage of the transformed values are considered at this stage. These results were written to a file, and then later read back in to calculate the amount of signal lost. The simulation expresses the precision in the form of 2 parameters: number of bits to the left of the radix point (indicating the integer part), and number of bits to the right of the radix point (indicating the fractional part). Note that an additional bit must be used for the sign.

The Daubechies wavelet with 4 coefficients are used in this study. First, the image was decomposed into 1 octave of resolution, and the parameters were changed to find the amount of precision needed to give perfect reconstruction. This indicates that there is no significant loss in the values stored in fixed-point format. This experiment ran several more times, for 2 and 3 octaves, and for all 9 test images.

This experiment finds the minimum number of bits needed to store transformed image data. When 8.3 bits (meaning 8 bits for the integer and 3 bits for the fraction) result in error, 9.2 would be tried, then 10.1, etc. The tables below give the resulting precision, depending upon the number of octaves, indicated in the first column. The next 2 columns of each table show the number of integer bits used followed by the number of bits used for the fractional part. The third column gives the summation of the error between the original image and the reconstructed image. Ideally, there should be no error, but storing the data with inadequate precision means that round-off will occur, which leads to error. The last column indicates the difference between the original and reconstruction in terms of the Peak Signal to Noise Ratio (PSNR).

4. RESULTS AND CONCLUSION

Tables 1 through 9 indicate how the precision affects the transform, namely that for perfect reconstruction, we need at least 13 bits (1+10+2) for 1 octave of decomposition, 14 bits (1+11+2) for 2 octaves, and 15 bits (1+12+2) for 3 octaves. Of course, more bits could be used to give a perfect reconstruction, for example, the precision used for 3 octaves would work fine for octaves 1 and 2. In determining the number of bits needed, most images were consistent in requiring fewer bits: 9.2 bits for 1 octave, 10.2 bits for 2 octaves, and 11.2 bits for 3 octaves. For the second octave, the maximum bits needed are 11.1 for one image, and 10.2 for the rest. To store all of them appropriately, 11 bits are needed for the integer part and 2 bits for the fractional part, resulting in 11.2 precision.

It is interesting to note that, for two additional runs of the experiment with the Dog image, the error produced between the original and reconstruction is close for 10.3 bits of precision (5685) and 11.0 bits of precision (5657). However, visible damage is noticeable for the first reconstructed image, but not the latter. Since the integer part has fewer bits in the 10.3 case, some high-order bits are lost. This means that the error for the 10.3 case is concentrated on a small area, while the 11.0 case has the error spread out. It is also important to note that adding more bits of precision does not guarantee a better result; a total of 14 bits were used in the 10.3 case, but the 11.0 case gives better results while using 2 fewer bits.

This experiment stores all transformed values with the same precision, regardless of the filters that generate them. The results indicate that the amount of precision necessary for the fractional part is not nearly as significant as it is for the integer part. One possible way to compact the signal in even less space is to use fewer bits to store the different DWT outputs. That is, store the output of the approximate signal with the precision indicated above, but use fewer bits to store the detail signals. This is a possible area for future study.

All of the data in the tables below assume a Daubechies wavelet with 4 coefficients. Changing the wavelet to the Daubechies 8-coefficient wavelet had the following effect: the same precision as shown in tables 2-4 are needed for 3 of the images (man, tank, and moon surface). In all other cases, another bit of precision is needed in 1 or 2 cases, though the total recommended does not change from above. That is, the Airplane, Jellybeans, Elaine and Lena images need 10.2 bits for 1 octave, the Airplane, Dog, and House images need 11.2 bits for 2 octaves, and the Dog, House and Lena images need 12.2 bits for 3 octaves.

This paper shows that the amount of precision (that is, the number of bits to keep after the DWT) is an important parameter behind the design of a system involving the wavelet transform. For a 2-D DWT, the number of bits to keep depends upon the levels of resolution. These tend to grow as the number of octaves increase. For the test images used in this study, 15 bits of precision allow the DWT outputs to be stored without affecting the reconstruction. One bit should be reserved for the sign (though it

would be possible to use 2's complement storage), 12 bits are needed for the integer part, and two bits are needed for the fractional part.

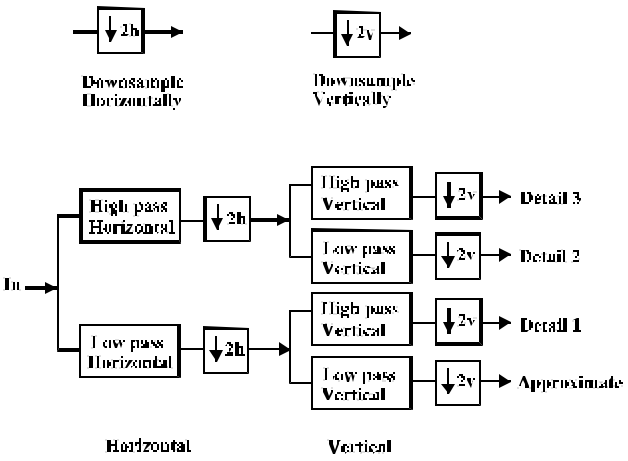

Figure 1 - The 2-Dimensional DWT

Figure 2 - Dog on Porch
This image is freely available at the web page http://carmaux.cs.gsu.edu

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	8	3	6959232	127
1	9	1	2	325
1	9	2	0	----
2	9	2	6776826	127
2	10	1	7	313
2	10	2	0	----
3	10	2	6663109	127
3	11	1	7	313
3	11	2	0	----

Table 1 - Precision for "Jellybeans" image

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	8	3	4769820	131
1	9	1	2	325
1	9	2	0	----
2	9	2	4648817	131
2	10	1	4	318
2	10	2	0	----
3	10	2	4495407	131
3	11	1	5	316
3	11	2	0	----

Table 2 - Precision for "Moon Surface" image

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	8	3	22853726	159
1	9	1	89	331
1	9	2	0	----
2	9	2	22866518	159
2	9	3	22863542	159
2	10	1	170	324
2	10	2	0	----
3	10	3	22677186	159
3	11	0	55530	265
3	11	1	144	325
3	11	2	0	----

Table 3 - Precision for "Man" image

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	8	3	24189120	142
1	9	0	24385	257
1	9	1	17	332
1	9	2	0	----
2	9	2	23984752	142
2	10	1	15	333
2	10	2	0	----
3	10	2	23810316	142
3	10	3	23808713	142
3	11	1	18	331
3	11	2	0	----

Table 4 - Precision for "Tank" image

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	8	3	26409964	141
1	9	0	21257	259
1	9	1	18	331
1	9	2	0	----
2	9	2	26592397	141
2	9	3	26589083	141
2	10	1	18	331
2	10	2	0	----
3	10	2	26249968	141
3	10	3	26248201	141
3	11	1	12	335
3	11	2	0	----

Table 5 - Precision for "House" image

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	8	3	7982370	125
1	9	1	2	325
1	9	2	0	----
2	9	2	7982370	125
2	10	0	4706	246
2	10	1	0	----
3	10	2	7805690	125
3	10	3	7805156	125
3	11	1	2	325
3	11	2	0	----

Table 6 - Precision for "Airplane" image

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	8	3	18497804	145
1	9	1	8	339
1	9	2	0	----
2	9	3	18344866	145
2	10	1	8	339
2	10	2	0	----
3	10	3	18006450	145
3	11	1	14	333
3	11	2	0	----

Table 7 - Precision for "Lena" image

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	8	3	19215519	144
1	9	1	9	338
1	9	2	0	----
2	9	2	19241387	144
2	10	1	14	334
2	10	2	0	----
3	10	2	19416442	144
3	11	1	13	334
3	11	2	0	----

Table 8 - Precision for "Elaine" image

Octaves	Integer bits	Fraction bits	S Error	PSNR
1	9	2	1434	206
1	10	1	1	330
1	10	2	0	-----
2	10	2	5685	192
2	11	1	0	-----
3	12	1	1	331
3	12	2	0	-----

Table 9 - Precision for "Dog" image

REFERENCES

[1] Andrew Bruce, David Donoho, and Hong-Ye Gao, "Wavelet Analysis," *IEEE Spectrum*, Oct. 1996, pages 26-35.

[2] Mohan Vishwanath and Chaitali Chakrabarti, "A VLSI Architecture for Real-Time Hierarchical Encoding/Decoding of Video using the Wavelet Transform," *IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP '94)*, Adelaide, Australia, Volume 2, April 19-22, 1994, pages 401-404.

[3] Keshab K. Parhi and Takao Nishitani, "VLSI Architectures for Discrete Wavelet Transforms," *IEEE Trans. VLSI Systems*, Vol. 1, No. 2, 1993, pages 191-202.

[4] Tinku Acharya, Po-Yueh Chen, and Hamid Jafarkhani, "A Pipelined Architecture for Adaptive Image Compression using DWT and Spectral Classification," *Proc. of the Thirty-First Annual Conf. on Info. Sciences and Systems*, Volume II, Baltimore, MD, March 19-21, 1997, pages 1-17.

[5] Stephane Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Patt. Anal. & Mach. Intel.*, Vol. 11, No. 7, 1989, pages 674-693.

[6] M. Vishwanath, R. M. Owens, and M. J. Irwin, "Discrete Wavelet Transforms in VLSI," *Proc. of the Int. Conf. on Application Specific Array Processors*, Berkeley, CA, August 1-2, 1992, pages 218-229.

[7] Gilbert Strang and Truong Nguyen, *Wavelets and Filter Banks*, Wellesley, MA, Wellesley-Cambridge Press, 1996.

[8] G. Knowles, "VLSI Architecture for the Discrete Wavelet Transform," *Electronics Letters*, Vol. 26, No. 15, 1990, pages 1184-1185.

[9] Charles C. Chang, Jyh-Charn Liu, and Andrew K. Chan, "On the Architectural Support for Fast Wavelet Transform," *SPIE Wavelet Applications IV*, Vol. 3078, Orlando, Florida, April 21-25, 1997, pages 700-707.

[10] Aleksander Grzeszczak, Mrinal K. Mandal, Sethuraman Panchanathan, and Tet Yeap, "VLSI Implementation of Discrete Wavelet Transform," *IEEE Trans. VLSI Systems*, Vol. 4, No. 4, 1996, pages 421-433.

[11] Jimmy Limqueco and Magdy Bayoumi, "A 2-D DWT Architecture," *Proc. 39th Midwest Symposium Circuits & Systems*, Ames, Iowa, August 18-21, 1996, pages 1239-1242.

[12] A. S. Lewis and G. Knowles, "VLSI Architecture for 2-D Daubechies Wavelet Transform without Multipliers," *Electronics Letters*, Vol. 27, No. 2, 1991, pages 171-173.

[13] Jun Wang and H. K. Huang, "Three-dimensional Medical Image Compression using a Wavelet Transform with Parallel Computing," *SPIE Imaging Physics*, San Diego, Vol. 2431, March 26-April 2, 1995, pages 16-26.

[14] *Wavelet Transform Processor Chip User's Guide*, Aware, Inc., Bedford, MA, 1994.

[15] Michael Weeks, Jimmy Limqueco, and Magdy Bayoumi, "On Block Architectures for Discrete Wavelet Transform," *32nd Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 1-4, 1998.

[16] Michael Weeks and Magdy Bayoumi, "3-D Discrete Wavelet Transform Architectures," *IEEE Int. Symp. Circuits & Systems (ISCAS '98)*, Monterey, CA, May 31 - June 3, 1998.