Reconfigurable Implementation of Wavelet Integer Lifting Transforms for Image Compression

S. L. Bishop, S. Rai, B. Gunturk, J. L. Trahan, and R. Vaidyanathan

Department of Electrical and Computer Engineering
Louisiana State University
Baton Rouge, LA 70803 (USA)
Email: sbisho4@lsu.edu, {suresh, bahadir, trahan, vaidy}@ece.lsu.edu

Abstract

Modern digital image processing requires powerful data compression algorithms to allow the data to be efficiently transferred from the host to end users (and back again). A typical 512×512 grayscale image of uncompressed data requires more than quarter of a million bytes. Current image compression standards like JPEG2000 and the FBI WSQ (wavelet scalar quantization) use wavelet transforms with quantization to compress still images, which reconstruct with high accuracy. This paper considers a number of popular 9/7 wavelet transform architectures. High level software models are developed for these transforms to validate their effectiveness. These software models are modified and evaluated as reversible integer wavelet lifting transforms. Further, using a virtual hardware design targeted to reconfigurable FPGA technology these transforms are implemented into a 2-D discrete wavelet transform (DWT) image processor with DDR SDRAM operating at core speeds of 200+ MHz. Finally, our Matlab and Maple models perform the validation of wavelet lifting transforms.

1. Introduction

Wavelets, today, find applications in a number of areas encompassing different aspects of signal processing. Some typical applications include denoising, edge detection, feature extraction, speech recognition, and echo cancellation. This paper applies wavelets in the area of image processing and discusses an FPGA-based implementation.

This work was supported by the National Science Foundation under grants CCR-0310916 and ECS-0528785. Authors would like to thank Xilinx for the kind donation of their Xilinx ISE software to our project.

In the Internet-driven world, an explosion of image data now permeates all aspects of our lives. Images for commerce, security, or just plain fun can tie up a substantial amount of bandwidth. For example, a single FBI 768×768 thumbprint represents 589824 pixels×8 bits/pixel bits of data. It's obvious that there is a need for quality image compression techniques. Early JPEG compression used 8×8 pixel blocks with 2-D discrete cosine transforms (DCT) for compression ratios of 15:1, but show loss of important high frequency information due to tiling artifacts (Figure 1(a)). The FBI research evolved from dissatisfaction with tiling artifacts into the wavelet scalar quantization (WSQ) standard [1], which implements fingerprint data compression using lossy subband wavelet coding, adaptive quantization, and variable length Huffman entropy coding, achieving compression ratios on the order of 18:1, with the reconstructed image showing very little image degradation (Figure 1(b)). The JPEG2000 [2] group in return adopted the use of wavelets, which when used along with arithmetic entropy coding, can achieve compression up to 200:1. Note wavelets offer a high compression capability and better resolution properties.

Because of the popularity of wavelets, it is imperative to explore hardware implementations. Hardware concepts such as pipelining and distributed arithmetic may help achieve better throughput. This paper explores one such implementation using Field Programmable Gate Arrays (FPGAs). This paper details our investigation into wavelet coding for compression of still images, and the implementation of lossless 5/3 and lossy 9/7 discrete wavelet transforms in reconfigurable hardware using the hardware description language Verilog. Note that FPGAs contain an array of cells and routing channels, which can be programmed to suit a specific application. Another advantage of using FPGAs is reprogrammability. As they do not have the arithmetic capabilities of general purpose digital signal

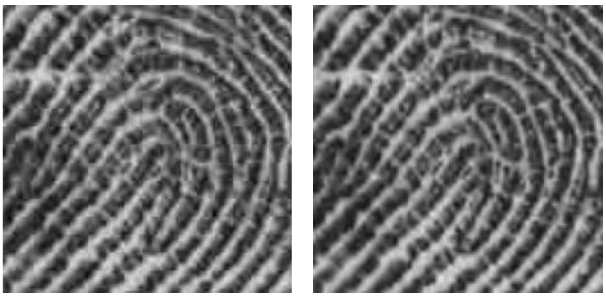

Figure1. Reconstructed images (a) Left – using DCT, (b) Right – using WSQ

processing chips, the implementation of wavelets is, thus, challenging.

Several researchers have used FPGA implementations for wavelets. Spiliotopoulos *et al*. [3] considered 9/7 filter obtained from both lifting and conventional design approaches and implements using multiplier-based and adder-based architectures. The authors studied quantization effects, and results show that the adder-based approach is faster as compared to multiplier-based. Barua *et al*. [4] also considered a multiplierless VLSI implementation of the 9/7 filter. Our approach is similar to this method, but it also considers integer-lifting, which helps optimize speed and area. The proposed FPGA hardware implements a pipelined 9/7 integer DWT/IDWT operating at 200 MHz, with additional control circuitry added to fully utilize an off chip 200 MHz double data rate (DDR) SDRAM, targeting 2-D image processing applications. It is capable of processing a 1-D N pixel image in $N/2$ clock cycles. Our proposed architecture also includes designing for easy modification to different interface protocols and memory sizes.

The layout of the paper is as follows. We discuss WSQ and JPEG2000, two popular compression formats, in Section 2. The 9/7 and 5/3 wavelet filters, used in the standards, help achieve lossy and lossless compression, respectively. We present basic ideas about wavelets in Section 3. Section 4 describes an FPGA implementation of 5/3 and 9/7 filters. It also gives various design decisions used in the implementation. We give some of our results in Section 5. Finally, Section 6 provides conclusions and future research.

2. Compression Formats

The FBI uses the wavelet scalar quantization (WSQ) method for encoding/decoding fingerprint images [1]. Each image is first normalized with zero mean and pixel value range (-128,128). It is then decomposed into 64 spatial frequency DWT subbands. Note that a

Figure 2. Subband structure of 2-D DWT

2-D decomposition of an image (Figure 2) shows the spatial relation across the four subbands: LL – DC or averaged subband, LH – Vertical subband, HL – Horizontal subband, and HH – Diagonal subband. Most of the signal energy is packed into the small area of the high-level LL-subbands. Low-level sub-bands have small amplitudes and little correlation. These 64 subbands come from 5 levels 2-D DWT. Each of the subbands is quantized using values from a quantization table based on the natural log of the variance of the inner block of pixels within each subband. The quantized coefficients are then passed to a Huffman encoder which compresses the data. The same tables specified for an encoder to use to compress a particular image are provided to a decoder to reconstruct that image.

The JPEG2000 still image compression standard is based on wavelet/subband coding techniques [2]. It uses both lossy (9/7) and lossless (5/3) transforms. To code transform data, the codec makes use of bitplane coding techniques. A linear transform is used to obtain a representation of the input image in a different domain. Quantization is carried out on the transformed coefficients in order to smooth out high frequencies that the human visual system cannot perceive, and transformed data is re-ordered so as to reduce the relative difference between quantized coefficients. This pre-processed stream is finally entropy coded using standard techniques, e.g., embedded block coding with optimized truncation, Huffman, and run-length.

3. Wavelets – Background

3.1 Orthogonal and Biorthogonal Filters

For non-stationary signals (such as image, speech, etc.), we need to know both frequency and time information simultaneously. The wavelet transform is suitable for such signals as it uses more than one analysis window. It allows data to be cut up into different frequency components and studied at those resolutions.

Wavelets are basis functions that decompose a signal into a linear superposition of dilations and translations of the scaling function $\phi(x)$ and the wavelet function $\psi(x)$.

Mathematically, the scaling (wavelet) function is $\phi(x)=\sqrt{2}\sum_n h_n\phi(2x-n)$ ($\psi(x)=\sqrt{2}\sum_n g_n\phi(2x-n)$) for a suitable set of low (high) pass filter coefficients h_n (g_n). Note that the translations and dilations of both $\phi(x)$ and $\psi(x)$ are orthonormal individually and also amongst each other. If $g_n=(-1)^n h_{M-n}$ (M is an odd number), the scaling and wavelet functions can be used to create perfect reconstruction (PR) filter banks. Let $\tilde{h}_n$ ($\tilde{g}_n$) be the low (high) pass filter coefficients of the analysis filter. For synthesis low (high) pass filter $h_n=\tilde{h}_n$ ($g_n=\tilde{g}_n$). The orthogonal filter, so designed, always has an even number of coefficients and is not symmetric (except for the trivial Haar transform).

For a more flexible framework, *biorthogonal wavelets* are used [5][6]. Considering $\tilde{h}_n$ ($\tilde{g}_n$) as the analysis low (high) pass filter and h_n (g_n) as the synthesis low (high). pass filter, the biorthogonal system is designed by quadrature mirroring the low pass filters

$$\tilde{G}(z)=z^{-1}H(-z),\ G(z)=z\tilde{H}(-z) \tag{1}$$

so that aliasing condition is achieved [5][6]. To satisfy the perfect reconstruction condition, we select the product filter $P(z)=\tilde{H}(z)H(z)$ with the additional constraint $P(z)+P(-z)=2z^{-d}$. It means the high pass filters are determined from the two low pass filters for the biorthogonal filter bank. Section 3.2 describes filter structures that help obtain analysis and synthesis low pass filters using the concept of product filter.

3.2 Filter Structures

A general form for a PR product filter $P(z)$ is that of an iterated filter bank [7]:

$$P(z)=(1+z^{-1})^{2p}\frac{1}{2^{2p-1}}\sum_{k=0}^{p-1}\binom{p+k-1}{k}(-1)^k z^{-(p-1)+k}(2)^{2k} \tag{2}$$

In (2), $(1+z^{-1})^{2p}$ represents a binomial spline filter and remaining factor is needed to guarantee perfect reconstruction. The $2p$ zeroes at z = -1 or $\omega=\pi$ are important for stability. It widens or "flattens" the frequency factorization, given in (3), yields the "9/7" transform where p = 4:

$$P(z)=(1+z^{-1})^{8}\frac{1}{2^{7}}\sum_{k=0}^{3}\binom{3+k}{k}(-1)^k z^{-3+k}\left(\frac{1-z^{-1}}{2}\right)^{2k} \tag{3}$$

Another method for factorizing a PR product filter is the Lagrange half band filter (LHBF) [8]:

$$P(z)=z^k\left(\frac{1+z^{-1}}{2}\right)^{2k}R_k(z) \tag{4}$$

Note, in (4), the $2k$ zeroes at π are important for stability, and the $R_k(z)$ factor is used to guarantee perfect reconstruction. The following substitution allows for factorizations that guarantee linear phase:

$$z\left(\frac{1+z^{-1}}{2}\right)^2\rightarrow\frac{1+Z}{2},\quad z^k\left(\frac{1+z^{-1}}{2}\right)^{2k}\rightarrow\left(\frac{1+Z}{2}\right)^k$$

$$P(Z)=\left(\frac{1+Z}{2}\right)^k\sum_{n=0}^{k-1}\binom{n+k-1}{n}\left(\frac{1-Z}{4}\right)^n$$

So the 9/7 filter with k = 4 can be represented as in (5)

$$P(Z)=\left(\frac{1+Z}{2}\right)^4\sum_{n=0}^{3}\binom{n+3}{n}\left(\frac{1-Z}{4}\right)^n. \tag{5}$$

In (4), $P(Z)$ is constrained by k zeros at Z = -1. If we reduce this number of zeros, we introduce some additional degrees of freedom with which we can modify the coefficients to power of 2 filter coefficients, and a free parameter with which we can tailor the filter characteristics to match the application, as first given by Tay [8]. Further manipulations of the zeros and gain factor of the product matrix have been used by Barua *et al*. [4] to create useful custom filter implementations optimized for particular applications, e.g., minimizing number of bits used in computation. The essence of this is applying limiting conditions based on using the stabilization zeros at z = -1 (Z = -1), and manipulating the remaining 4 zeros and gain to try to make the product filter output look like the original general form for a PR product filter as discussed earlier.

4. Design Decisions

4.1 Lifting Architecture

There are various architectures for implementing a two channel filter bank. Note that a filter bank consists of a low- and high-pass filter and decimator and expander. Four structures, namely, direct, polyphase, lattice, and lifting are considered with reference to discrete wavelet transform. The direct form provides a straightforward computation but lacks efficiency. Polyphase analysis improves efficiency by 50% by avoiding

redundant computations and is suitable for hardware realization. The lattice and lifting structures further improve efficiency by reducing the number of computations performed. This paper implements the lifting structure.

References [9] [10] proposed the lifting technique, which allows the development of integer-only wavelet and scaling coefficients. The *forward lifting* transform uses three steps: *split* divides the input into disjoint even and odd samples, *predict* uses even-indexed samples to obtain the difference or detail coefficient, and *update* provides the smoothing (or coarser) coefficient from even samples and detail. The inverse lifting transform is obtained directly by reversing the forward transform, i.e., change the order of operations, invert the signs in the lifting steps, and replace the data split with a data *merge*.

A polyphase formulation of the lifting steps plays an important role in constructing higher order wavelets starting from the simple initial wavelet (such as a *lazy* wavelet). It is because the analysis polyphase matrix $P_a(z)$ of any biorthogonal wavelet of compact support can be factored into a diagonal matrix and several pairs of lifting $\begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix}$ and dual lifting steps $\begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix}$, where $s_i(z)$ and $t_i(z)$ are Laurent polynomials.

$$P_a(z)=\begin{bmatrix} \tilde{H}_e(z) & \tilde{H}_o(z) \\ \tilde{G}_e(z) & \tilde{G}_o(z) \end{bmatrix}=\begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix}\prod_{i=1}^{m}\begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix};\ K_1,K_2\neq 0$$

Similarly, using this concept, the inverse transform is achieved by changing all the signs of polynomials in the forward transform, and then running it backwards.

For the biorthogonal 9/7 tap filter, the analysis (synthesis) filter has 9(7) coefficients. Using the symmetry property of coefficients of the $\tilde{h}-$ filter, the even and odd terms of $\tilde{H}(z)$ can be expressed as $\tilde{H}_e(z)=h_4\left(z^2+z^{-2}\right)+h_2\left(z+z^{-1}\right)+h_0$ and $\tilde{H}_o(z)=h_3\left(z^2+z^{-1}\right)+h_1\left(z+1\right)$, respectively. A particular factorization, called CDF (Cohen-Debauchies-Faveau), is symmetric, where every quotient is a multiple of $\left(1+z\right)$.

$$P_a(z)=\begin{bmatrix} \tilde{H}_e(z) & \tilde{H}_o(z) \\ \tilde{G}_e(z) & \tilde{G}_o(z) \end{bmatrix}=P_1(z)P_2(z)K$$

Note that

$$P_1(z)=\begin{bmatrix} 1 & \alpha\left(1+z^{-1}\right) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \beta\left(1+z\right) & 1 \end{bmatrix}$$

$$P_2(z)=\begin{bmatrix} 1 & \gamma\left(1+z^{-1}\right) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \delta\left(1+z^{-1}\right) & 1 \end{bmatrix},\ K=\begin{bmatrix} \xi & 0 \\ 0 & 1/\xi \end{bmatrix}$$

Figure 3 illustrates the implementation where d (s) represents difference (sum) corresponding to a detail (smooth) term in the lifting step.

For applications like WSQ/JPEG2000, the grayscale data is represented as 8-bit binary numbers. The lifting scheme can be easily modified to map integers to integers [10], while still maintaining the PR conditions. In general, predict and update values are not integers. If we round these numbers to the integers, then

$$d_j^{new}=d_j-\left\{P\left(s_j\right)\right\}\qquad s_j^{new}=s_j+\left\{P\left(d_j\right)\right\},$$

where we use curly braces to represent the rounding operation (e.g., floor, ceil, nearest integer). Since the rounding operation doesn't change the way the value inside is calculated, then we can perfectly reconstruct, i.e.,

$$d_j=d_j^{new}+\left\{P\left(s_j\right)\right\}\qquad s_j=s_j^{new}-\left\{P\left(d_j\right)\right\}.$$

$$s_l^{(0)}=x_{2l}\qquad d_l^{(0)}=x_{2l+1}$$
$$d_l^{(1)}=d_l^{(0)}+\alpha\left(s_l^{(0)}+s_{l+1}^{(0)}\right)\qquad s_l^{(1)}=s_l^{(0)}+\beta\left(d_l^{(1)}+d_{l-1}^{(1)}\right)$$
$$d_l^{(2)}=d_l^{(1)}+\gamma\left(s_l^{(1)}+s_{l+1}^{(1)}\right)\qquad s_l^{(2)}=s_l^{(1)}+\delta\left(d_l^{(2)}+d_{l-1}^{(2)}\right)$$
$$s_l=\xi s_l^{(2)}\qquad d_l=d_l^{(2)}/\xi$$

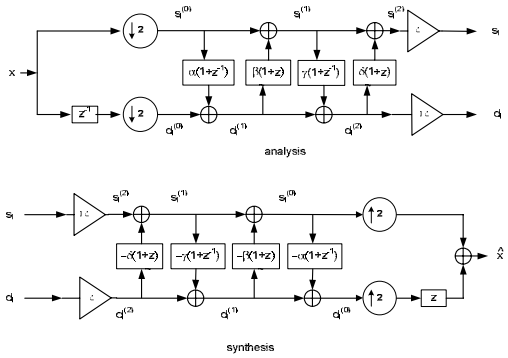

Figure 3. Lifting implementation for 9/7 transform

4.2 Quantization

The 9/7 multiplication constants are irrational numbers, which can be represented using floating point data. Floating point hardware is in general complex, involving large area on the silicon die, and for image processing applications it is much slower than a fixed

point (FP) representation. However, limiting the number of bits using a fixed point implementation introduces truncation error. In image processing the mean square error (MSE) and the peak signal-to-noise ratio (PSNR) are often used to quantify this error. Given original image $I(i, j)$ N pixels wide by M pixels tall, and reconstructed image $I'(i, j)$

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[I(i,j) - I'(i,j)]^2}{MN} \text{ and}$$

$$PSNR = 10\log_{10}\left(\frac{(255)^2}{MSE}\right)$$

What is needed is a PSNR that maintains the desired human subjective perception. This allows some room to tailor the DWT/IDWT data path bit width and coefficient accuracy, which translates into smaller memory and hardware requirements. For the 9/7 transform the largest coefficient is α = -1.586134342. For a gray-scale image, each pixel is of 8-bit size. Thus

$$\lceil pixel + 1.58*(pixel + pixel)\rceil = \lceil 2^8 + \lceil 1.58*2*2^8\rceil\rceil = 2^{11}$$

Including the multiplier $1/\xi$ in the IDWT requires another bit. If we map integers to integers, then our worst case requires another 12 bits be added to include all possibilities, but the multiplied terms can be truncated by using, for example, the floor function, to an integer value. Since interconnects we investigated such as Universal Serial Bus (USB) [11] and DDR SDRAM can transfer data in 16-bit units, this gives 3 bits of safety for integer-to-integer calculations. Matlab simulations using 7 different standard B&W test images verified 16-bits to be sufficient.

The hardware complexity can be further reduced by using shift-add operations and Canonic Signed Digit Arithmetic (CSDA) [4] to represent the multiplication by the constant coefficients. CSDA allows encoding of a binary number so that it contains the fewest number of non-zero bits, with no two consecutive bits in a CSD number being non-zero, i.e.,

$$x = \sum_{r=0}^{N-1} s(r)2^r \quad ; \quad s(r) \in \{-1, 0, 1\}.$$

All internal calculations for the 3 stages of the DWT/IDWT are made using 16-bit resolution for the fractional approximations for the constant coefficients, requiring up to 32-bit width adders/subtractors in the data path. Figure 4 shows an example of the Barua *et al* [4] lifting multipliers expressed in the CSDA form. We implement shift-add multiplier using this concept.

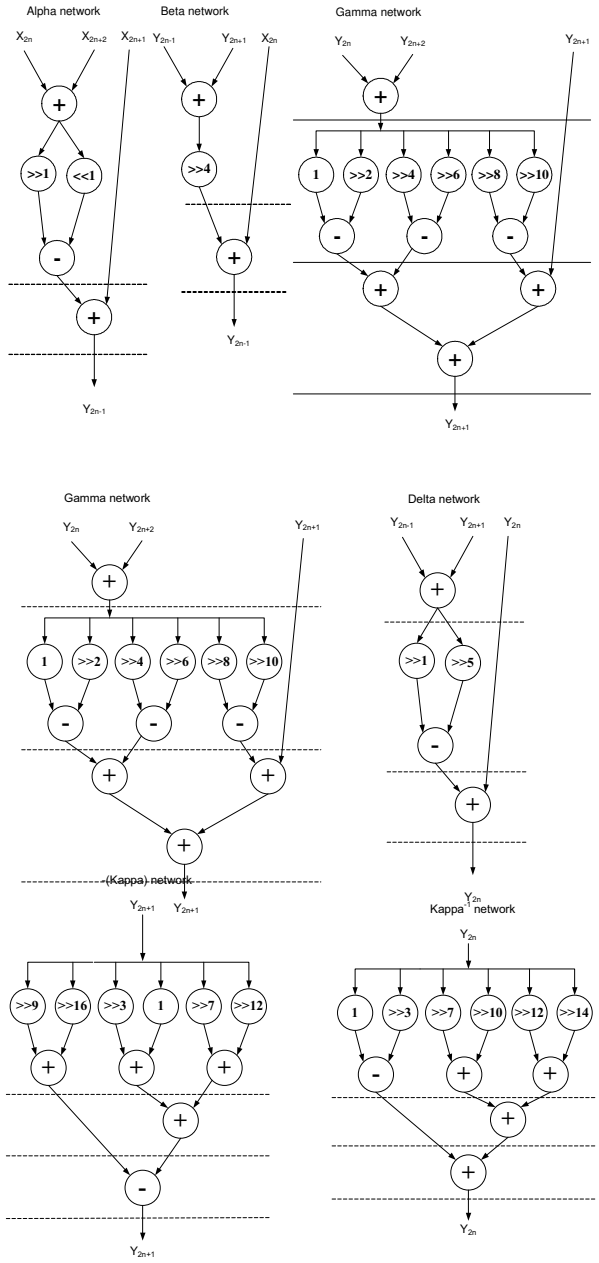

Figure 4. Canonic signed digit multiplier example

4.3 Virtual Hardware Modeling\Verification

In order to design a complex digital device such as a discrete wavelet transform, we use the hardware description language (HDL) Verilog. An early design decision was to opt for redundant hardware to minimize shared resources, and pipelining wherever reasonable to allow the chip to be run at a higher clock speed. Design decisions were also made to accommodate both the initial USB interface for communication between the DWT/IDWT engine and the CPU, DDR

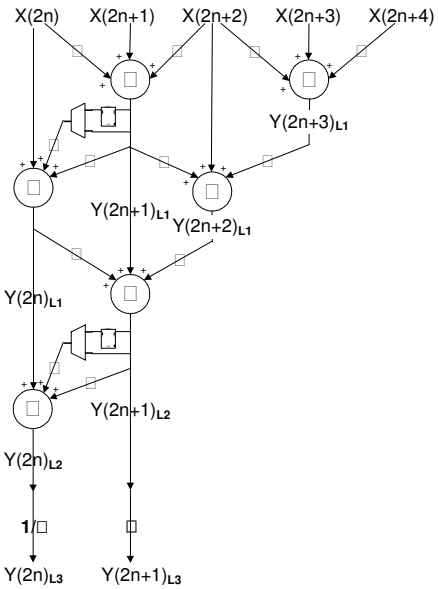

Figure 5. 9/7 DWT data path

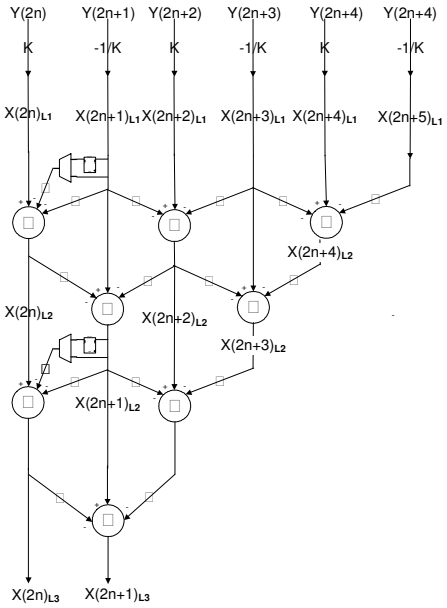

Figure 6. 9/7 IDWT data path

SDRAM data widths {16, 32, and 64} bits, and for its future additions PCI and Ethernet interface.

For the large image sizes required for say WSQ compliance, even the largest Virtex-4 total block RAM (BRAM) cannot hold the required max size 1600×1000 16-bit pixel images. One solution investigated was to use a double data rate synchronous dynamic random access memory (DDR SDRAM) to hold the image data during each pass through the DWT/IDWT, and to the previously mentioned universal serial bus (USB) interface to communicate with the CPU. With 8-byte data width and a bus frequency of 200 MHz, DDR-SDRAM gives a max transfer rate of 3200 Mbytes/s. In terms of maximum sized WSQ images we have 1000 images/s. This high rate of data of data transfer allows for the possibility of real-time processing. Figure 5(Fig. 6) shows the 9/7 3-stage cascaded filter data path required to do the calculations for the forward (inverse) discrete wavelet transform.

After an initial 10 cycle latency our proposed architecture can continuously produce an odd/even pair of 16-bit data for addresses ($2n$, $2n$+1) at clock rates near 200 MHz. Another area for improvement is the transfer of the data to/from the DDR SDRAM. WSQ and JPEG2000 formats require multiple level 2-D DWT/IDWT transformations. A 2-D DWT requires rastering each line of an M row by N column image through the 1-D DWT, splitting the line into even/odd components {0,2,4,…,N-2,1,3,5,…,N-1}, and transposing into an $N\times M$ image, and then repeating this 3-step process to return the transformed $M\times N$ image. A 2-D IDWT requires first splitting each line of an M row by N column image {0,N/2+1,1,N/2+1,…,N/2,N-1}, rastering each line of the image through the 1-D DWT/IDWT, }, and transposing into an $N\times M$ image, and then repeating this 3-step process to return the recovered original $M\times N$ image. The splitting is handled using a shuffle network. In order to fully utilize the burst capability of the DDR SDRAM, a transpose network feeds eight 2048x16 dual port rams, allowing data transfers of 128-bit to the DDR SDRAM. Since the DWT image data was written into the DDR SDRAM transposed, it can be burst read out as a raster line. The FPGA platform allows for rapid prototyping of different configurations. Figures 7 and 8 provide two different architectures developed to use a USB interface to communicate with the CPU.

The Xilinx Integrated Synthesis Environment (ISE) environment performed the synthesis, translation, place and route, and mapping. The ISE allows both GUI and command line operations, so initial GUI work was leveraged to create synthesis/PAR scripts. The multiple clock domain synchronization issues and pin-out details needed for timing constraints and floor-planning, resp.; were supplied to the router using the user constraints file (UCF).

5. Results and Discussion

The validity of the wavelet lifting transforms are verified by first modeling them using Matlab and Map-

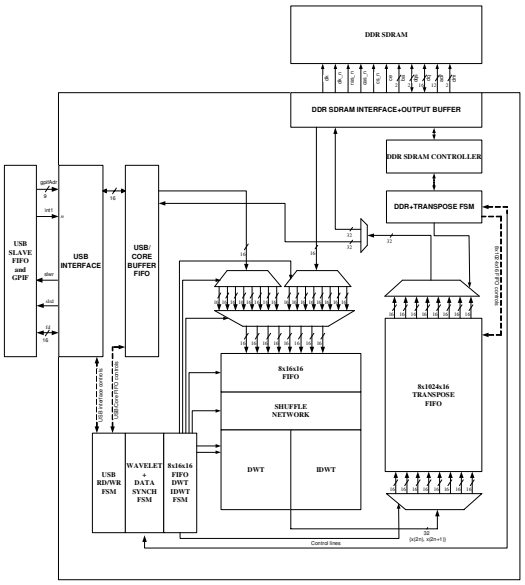

Figure 7. DWT/IDWT prototype #1

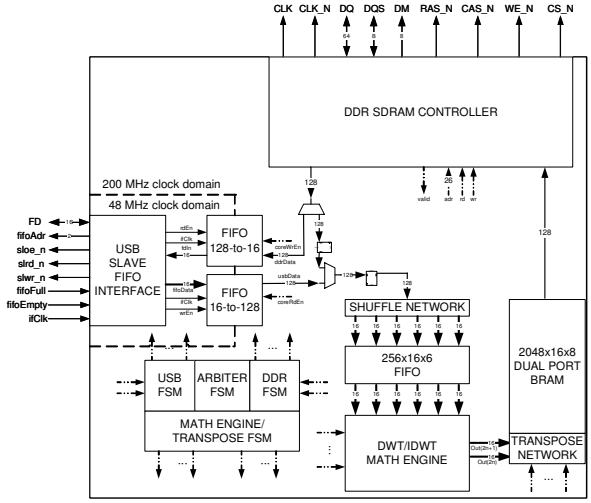

Figure 8. DWT/IDWT prototype #2

Table1. Sample PSNR for WSQ rational, integer, and 16-bit hardware reconstructions

Image	CDF-1 PSNR rational (dB)	CDF-1 PSNR integer (dB)	Tay-3 PSNR rational (dB)	Tay-3 PSNR integer (dB)	Tay-3 PSNR 16-bit integer (dB)
lena	35.179	34.985	27.956	23.489	23.489
finger	38.352	38.872	24.071	26.028	26.028
barbara	33.582	33.845	24.455	23.271	23.271
mandrill	33.476	33.583	22.972	23.494	23.494
peppers	32.905	33.050	33.951	22.348	22.348
boat	39.373	39.771	26.895	29.210	29.210
goldhill	36.837	37.219	27.958	25.557	25.557

ple. Calculations for lifting coefficients were performed using Maple's symbolic math engine. The numeric calculations were handled using Matlab. This allowed the freedom to try different filters for the lifting transforms, to determine the quality of results, prior to implementing in hardware.

Results were calculated for a variety of classic black and white image processing photos using the integer transform versions Cohen-Debauchies-Faveau (CDF) 9/7 transform as a single filter (CDF-1) and 3-stage cascaded filter (CDF-3), Tay 9/7 3-stage cascaded filter (TAY-3), and the Carletta-Kotteri-Bell (CKB) 9/7 3-stage cascaded filter (CKB-3) for a single, quadruple, and WSQ format 2-D DWT/IDWT using Huffman and RLE compression techniques. If we use the integer lifting transform with a "floor" function, which simplifies the hardware implementation, we observe an average PSNR drop of 0.49 dB. Limiting the TAY-3 DWT and IDWT transforms coefficients to 16 bits yielded the same PSNR as Matlab integer transforms using the floor function (refer to Table 1).

Table 2. WSQ-compliant 2-D DWT/IDWT using Huffman+RLE compression techniques

Image	CDF-1 PSNR (dB)	CDF-1 COMP RATIO	CDF-3 PSNR (dB)	CDF-3 COMP RATIO
lena	35.4018	6.3395	21.6822	8.5303
Finger	36.6987	8.3688	21.6807	9.6523
barbara	33.105	6.0794	20.7398	7.547
mandrill	33.3953	2.6947	21.0349	3.3068
peppers	33.0145	8.0551	20.0479	6.3263
boat	39.099	7.0011	22.8256	5.9887
goldhill	36.7581	5.4666	21.2469	4.0639

Image	TAY-3 PSNR (dB)	TAY-3 COMP RATIO	CKB-3 PSNR (dB)	CKB-3 COMP RATIO
lena	23.2674	13.9231	22.1467	123.2459
Finger	23.5741	9.9441	22.0304	105.0443
barbara	22.1261	8.8532	21.1047	140.4845
mandrill	22.5652	3.8448	21.5488	110.2372
peppers	21.4654	15.1932	22.3369	107.8782
boat	25.7009	9.7455	22.0889	124.0038
goldhill	23.524	12.7614	20.6972	138.4807

The modified TAY-3 and CKB-3 16-bit integer implementations in general show much greater compression of the various images, at the cost of image clarity as indicated by the reduced PSNR (Table 2). No trans-

Table 3. DWT with USB interface and DDR SDRAM controller map data

Device Utilization Summary		
Number of SLICEs	1907 out of 9280	20 %
Minimum period (core+ddr sdram controller)	4.973 ns	
Maximum Frequency (core+ddr sdram controller)	201.092 MHz	
Total estimated power consumption (Xpower):	861 mW	

Table 4. Performance comparison with existing FPGA model implementations

model	FPGA Used	Decomp. Levels	Slices	Freq (MHz)
[12]	Virtex II	2	985	50
[9]	APEX20K E	2	7726	66.8
[13]	4052XL	2	785	85.49
[14]	XCV300	2	853	89.1
[15]	XCVE2000	2	1402	159.51
CKB-3 integer	XCV2P20	2	1907	201.092

form outperforms the others for all images, indicating the information content of the images can be used to tailor the transforms for optimal performance.

The Xilinx XPower utility, run using Verilog value change dump (vcd) files from behavioral simulations, yielded an estimated power calculation of 861 mW (Table 3). The fully mapped prototype design #1 onto the targetted Virtex-2 Pro2vp20ff1152-7 FPGA die can be seen in Figure 9. Table 4 provides a comparison of previously published DWT/ IDWT implementations with our pipelined version.

6. Conclusions

Several methods proposed for designing 9/7 wavelet transforms are investigated. We have used high level math programming software like Matlab and Maple to verify different filter schemes. Utilizing the hardware description language Verilog, we created and verified the design of different discrete wavelet filters, including integer lifting scheme. Our shift-add multiplier uses CSDA concept given in [4]. The Xilinx ISE virtual hardware design platform provided quick feedback to the design decisions we made, allowing us to develop a 200 MHz USB interfaced 2-D DWT/ IDWT DDR SDRAM engine targeting a Virtex-2 Pro 2vp20ff1152-7 FPGA die. Reconstructed images from the wavelet engine suffer from loses of information due to corner effects and some bleaching of the image (rows 3 and 4 in Figure 10), but overall the high frequency details seem to have been retained, and PSNR are in line with

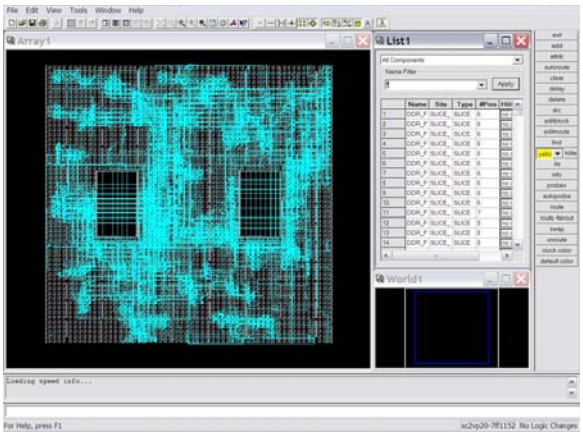

Figure 9. Placed and routed DWT/IDWT with USB interface and DDR SDRAM controller

previous work reported in this field. The compression ratios attained using the integer discrete wavelet transforms validate them as a tool for efficiently transferring digital image data. Investigation of machine learning algorithms to aid the search for optimal wavelet coefficients is an area for future research. Completion of all hardware WSQ and JPEG2000 solutions, with additional interfaces for standard protocols PCI and Ethernet, for real time solutions in data compression, as well as super-resolution image registration, are also being planned for future work.

7. References

[1] J. Bradley and C. Brislawn, "The FBI wavelet/scalar quantization standard for gray-scale image compression," Visual Info. Process. II, Volume 1961 of Proc. SPIE, pages 293–304, Orlando, FL April 1993 SPIE.

[2] M.D. Adams, "The JPEG-2000 Still Image Compression Standard," ISO/IEC JTC 1/SC 29/WG.

[3] V. Spiliotopoulos et al., "Quantization effect on VLSI implementations for the 9/7 DWT filters," preprint.

[4] S. Barua, et al., "An efficient architecture for lifting-based two-dimensional discrete wavelet transforms," INTEGRATION the VLSI journal, 38, pp. 341–352, 2005.

[5] M. Oslick, I. Linscott, S. Maslakovic, and J. Twicken, "A General Approach to the Generation of Biorthogonal Bases of Compactly-Supported Wavelets," Proc. IEEE Int'l Conf. Acoustics and Signal Processing, pp. 1537-1540, 1998.

[6] A.Cohen, I.Daubechies, and J. Feauveau, "Biorthogonal bases of compactly supported wavelets," Comm. in Pure and Appl. Math., vol. 45, 485-560, 1992.

[7] G. Strang, “Creating and Comparing Wavelets,” Dept. of Mathematics, MIT, pp. 5-9.

[8] D.B.H. Tay, "A class of lifting based integer wavelet transform, "2001 Int.l Conf. on Image Processing, Vol. 1, p. 602.

[9] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," SIAM Journal on Mathematical Analysis, Volume 29, Number 2, pp. 511-546, 1998.

[10] A.R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Lossless image compression using integer to integer wavelet transforms," Journal Proceedings of the Int. Conf. on Image Processing, Vol. I, pp. 596-599, Oct. 1997.

[11] “Universal Serial Bus Specification,” Copyright © 2000

[12] G.Kuzmanovs, et al., “Reconfigurable DWT unit based on lifting,” 12th Annual Workshop on Circuits, Systems, and Signal Processing (ProRISC), November 2002.

[13] S. Masud and J.V. McCanny, “Rapid design of biorthogonal wavelet transforms,” IEE Proceedings of Circuits, Devices and Systems, Volume: 147 Issue: 5, 2000, pp. 293 - 296.

[14] A. Al-Haj, “Fast Discrete Wavelet Transformation Using FPGAs and Distributed Arithmetic,” International Journal of Applied Science and Engineering, 2003, 160-171.

[15] I. Uzun and A Amira, “Design and FPGA implementation of high-speed discrete biorthogonal wavelet transforms,” School of Comp. Science, The Queen’s University Belfast.

Figure 10. Comparison of standard image sequences using WSQ (rows from top to bottom): originals, Cohen-Debauchies-Feauveau single stage floating point, CKB 9/7 3-stage integer, TAY 9/7 3-stage integer.