Poly-DWT: Polymorphic Wavelet Hardware Support for Dynamic Image Compression

AMIT PANDE and JOSEPH ZAMBRENO, Iowa State University

Many modern computing applications have been enabled through the use of real-time multimedia processing. While several hardware architectures have been proposed in the research literature to support such primitives, these fail to address applications whose performance and resource requirements have a dynamic aspect. Embedded multimedia systems typically need a power and computation efficient design in addition to good compression performance. In this article, we introduce a Polymorphic Wavelet Architecture (Poly-DWT) as a crucial building block towards the development of embedded systems to address such challenges. We illustrate how our Poly-DWT architecture can potentially make dynamic resource allocation decisions, such as the internal bit representation and the processing kernel, according to the application requirements. We introduce a filter switching architecture that allows for dynamic switching between 5/3 and 9/7 wavelet filters and leads to a more power efficient design. Further, a multiplier-free design with a low adder requirement demonstrates the potential of Poly-DWT for embedded systems. Through an FPGA prototype, we perform a quantitative analysis of our Poly-DWT architecture, and compare our filter to existing approaches to illustrate the area and performance benefits inherent in our approach.

Categories and Subject Descriptors: C.3 [**Special-Purpose and Application-Based Systems**]: *real-time and embedded systems*; I.4 [**Image Processing and Computer Vision**]: Compression (Coding)—*approximate methods*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Polymorphic architecture, wavelet transform, filter design

ACM Reference Format:
Pande, A. and Zambreno, J. 2012. Poly-DWT: Polymorphic wavelet hardware support for dynamic image compression. ACM Trans. Embedd. Comput. Syst. 11, 1, Article 6 (March 2012), 26 pages.
DOI = 10.1145/2146417.2146423 http://doi.acm.org/10.1145/2146417.2146423

1. INTRODUCTION

Multimedia services over embedded devices are becoming popular with the development of scalable encoders and rise of reconfigurable hardware to support the required high throughput. The large computational complexity and memory requirements involved in real-time image processing algorithms have been a bottleneck for such systems.

The Discrete Wavelet Transform (DWT) has emerged as a powerful tool for compression and is being used in many multimedia and signal processing applications. It constitutes a significant part of the overall computations involved in image/video compression schemes. Many image compression schemes have been derived from DWT-based structures [Said and Pearlman 1996; Shapiro 1993; Taubman 2000]. The work on using Embedded Zero-tree Wavelet (EZW) structures [Shapiro 1993] for image compression

Authors' addresses: A. Pande and J. Zambreno, Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011; email: {amit, zambreno}@iastate.edu.

DOI 10.1145/2146417.2146423 http://doi.acm.org/10.1145/2146417.2146423

was a milestone research that introduced sub-band coding to achieve high-compression performance. Said and Pearlman [1996] introduced a more efficient DWT-based Set Partitioning in Hierarchical Trees (SPIHT) encoding to improve the performance of Shapiro's EZW coding. [Taubman 2000] proposed the DWT-based Embedded Block Coding with Optimal Truncation (EBCOT) coding algorithm, which was accepted for scalable encoding in JPEG2000 [Christopoulos et al. 2000]. JPEG2000 achieves almost twice as much compression as JPEG with the same reconstruction quality of images (in terms of PSNR or Peak Signal-to-Noise Ratio). DWT has been incorporated in recent image and video compression research such as motion JPEG2000 [Qiu and Yu 2001]; 3-D , 4-D sub-band coding [Choi and Woods 1999; Yang et al. 2006]; and MPEG-4 SVC (Scalable Video Coding extension, released in July 2007) [Schwarz et al. 2007]. Of the fourteen proposals for SVC received by the MPEG committee, twelve were based on DWT, while two were extensions of the existing DCT based MPEG-4 AVC standard. Thus, DWT is increasingly becoming a popular choice for image/video compression due to high compression, scalability, and other features.

We recognize that DWT serves as backbone for new generation multimedia compression schemes and present a polymorphic architecture for its hardware implementation in this paper. Implementations such as those using ASICs or FPGAs are capable of accelerating these computations by exploiting the inherent algorithmic parallelism. Stroobandt et al. [2004] discuss the performance requirements of a reconfigurable hardware architecture for a scalable wavelet-based video decoder. Eeckhaut et al. [2007] present a complete video deliver chain including a video server, negotiation and scalable video clients using a wavelet-based coding scheme at its core. Many hardware implementations have also been proposed in the research literature [Alam et al. 2003; Benkrid et al. 2003, 2001; Ritter and Molitor 2001; Kotteri et al. 2005; Martina and Masera 2005; Villasenor et al. 1995]. These implementations aim at reducing hardware complexity in order to improve the system throughput.

Another concern is the fact that many typical applications of DWT have dynamic resource requirements. For example, consider a distributed video surveillance setup [Verma et al. 2006, 2008]. There are low-activity (idle) times and high-activity (active) times associated with the camera. During idle times, a low-power and low-bandwidth design may satisfy the requirements. However, during active times, the system would require transmission of a higher quality signal over a potentially sparse resource network. In such cases, it would be extremely beneficial to be able to distribute the available hardware resources to allow a compression efficient implementation using a relatively large amount of power.

In this article, we introduce a new layout and reconfiguration scheme for multimedia applications, which we call the Polymorphic Wavelet Architecture (Poly-DWT). We define *polymorphism* as the capacity of an architecture to adapt its hardware usage to meet the desired dynamic specifications. In the image processing domain, these specifications would be in terms of throughput, reconstruction quality, and power consumption, among others. Our Poly-DWT architecture allows the individual processing kernels to modify their hardware resources to suit the instantaneous application requirements. At its highest level, the Poly-DWT provisions for optimal device usage under the given performance and quality requirements. A fine-grained description of poly-DWT has been provided that allows run-time reconfigurability of the design over commodity FPGA platforms and ASIC designs.

Figure 1 gives a general description of Poly-DWT and its interface with a larger multimedia system. Some multimedia input (such as a stream of pixels for consecutive frames of a video) is first transformed into the time-frequency domain by the wavelet transform (DWT). Depending upon the throughput required and the input available from the video device, various instances of DWT kernels can be used in the

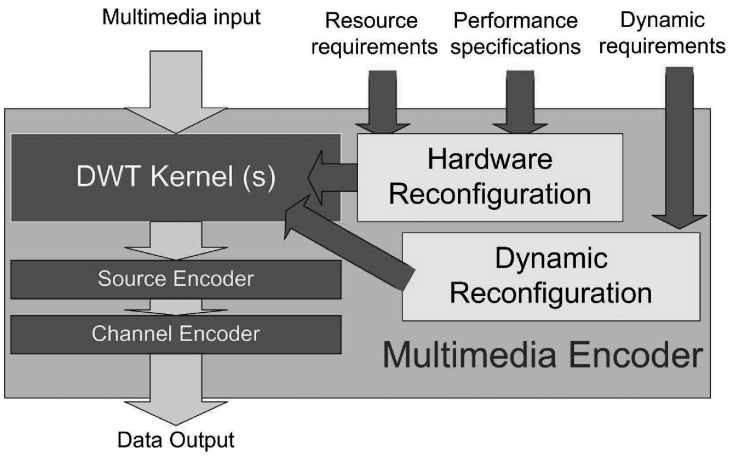


Fig. 1. Conceptual overview of the polymorphic wavelet architecture.

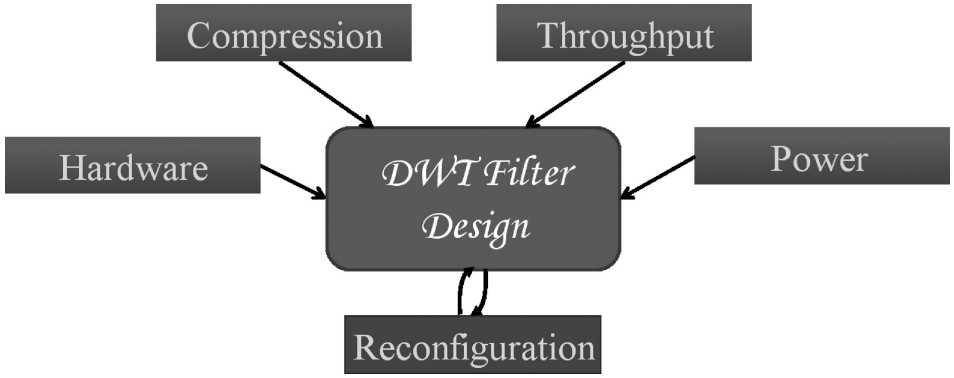


Fig. 2. Conceptual overview of the DWT filter design constraints and desired features.

implementation. The DWT kernel can be implemented using varying lengths, leading to varying image compression properties of the DWT block. An interface is provided for the application to dynamically notify the architecture about its performance requirements in terms of the hardware requirements and the image reconstruction quality requirements. Besides the previously mentioned video surveillance application, other real-time video streaming applications such as those used in medical image processing [Leeser et al. 2004], remote laboratories [Mittal et al. 2007], or educational video streaming [Pande et al. 2007] may benefit enormously from the polymorphism of DWT kernels.

We summarize the requirements of embedded multimedia system design in Figure 2 and they are enumerated as following.

—Modern embedded multimedia systems would require transmission of a high-quality signal over a potentially sparse resource network. Thus, good compression is a desired feature of an efficient implementation.
—High-system throughput and good perceptual quality are desired features and pose constraints on system design.
—Embedded systems have hardware and power constraints because they are typically mobile, battery-driven devices.

—Hardware reconfiguration of the filters is the enabling technology to realize these tradeoffs. Intelligent allocation of hardware resources can achieve a run-time tradeoff between hardware resources and performance constraints.

Our Poly-DWT architecture takes these explicit run-time requirements, along with an output feedback of the available hardware resources and image reconstruction quality and continually makes a reconfiguration decision. The reconfiguration mentioned in this article refers to the ability of our hardware to reconfigure its hardware resources. The implementation of our design can be done over FPGA, and ASICs. Given an image quality constraint the architecture can self-reconfigure to maximize device performance or power consumption, and given an external resource or performance constraint it can reconfigure to maximize image quality. Initial results had been presented in Pande and Zambreno [2008a, 2008b]. The proposed approach can provide a set of solutions for the dynamic requirements of system performance and power consumption without any overheads in throughput or hardware cost in comparison with existing approaches.

The contributions of this article can be summarized as follows.

—We introduce the concept of the Polymorphic Discrete Wavelet Transform (Poly-DWT) architecture. The Poly-DWT architecture enables dynamic allocation of hardware resources to efficiently create a dynamic response to changing external conditions.
—We discuss the development of a family of parameterized bi-orthogonal 9/7 filters and the derivation of binary coefficient filters for hardware efficient implementation.
—A multiplier-free binary 9/7 wavelet filter is introduced to obtain a faster and more efficient implementation.
—A switching scheme to allow runtime switching between 5/3 and 9/7 wavelet structures with hardware reuse is presented.
—We present a quantitative analysis of the various factors and tradeoffs involved in a Poly-DWT implementation.
—We present a detailed area/performance tradeoff analysis for the sample Poly-DWT filters.

The remainder of this article is organized as follows: In section 2, we give a working knowledge of DWT filters used in image compression. Section 3 provides a background study of the DWT algorithm and its hardware implementation. We also present the related research and limitations with existing hardware implementations. This motivates us for a Polymorphic design, which is presented in Section 4. The subsections give the mathematics, numerical study and background of the design. Next, we arrive at the candidate filters and their hardware architectures and then choose the optimal filters for Poly-DWT. Section 5 gives an insight into hardware re-allocation by changing the internal word width representation of registers by "bit-width" switching scheme. Section 6 introduces the "on-the-fly" switching scheme for filter coefficients. Section 7 also gives details of other experiments both in ModelSim and Xilinx ISE for hardware performance and over MATLAB for rigorous image processing performance measurements. In Section 8, we conclude the article with a look towards planned future work.

2. MOTIVATION AND INSIGHT

Prior works in signal processing explain that the 1-D DWT can be viewed as a signal decomposition using specific low pass and high pass filters. A single stage of image decomposition can be implemented by successive horizontal row and vertical column wavelet transforms. Thus, one level of DWT operation is represented by filtering with high- and low-pass filters across row and column successively and is illustrated in Figure 3. After each filtering a down sampling is done by a factor of 2 to remove the redundant information. The two most common DWT filters used in image compression

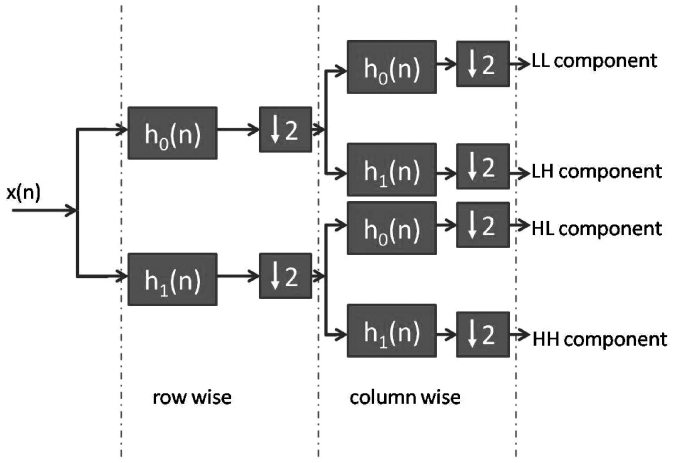


Fig. 3. Basic stages of a one level 2-D wavelet transform operation.

are Le Gall's 5/3 filter and the Daubechies 9/7 filter [Christopoulos et al. 2000]. They are accepted in the JPEG2000 standards. The Le Gall's filter has rational coefficients and its hardware implementation requires less resources. The Daubechies 9/7 (also commonly known as CDF 9/7) filter has better compression performance. However, it has irrational coefficients therefore its hardware requirements are very large.

This article develops the Poly-DWT architecture to serve as a backbone for real-time multimedia applications to address their dynamic demands and constraints. In this article, we discuss some dimensions that provide this polymorphism to our architecture. The first dimension is the choice of suitable DWT filter. Different applications such as medical image processing, High Definition Television, video surveillance applications, and wireless video all have different real-time constraints [Pande et al. 2007; Mittal et al. 2007] and different filters may serve the requirements at different times.

The complexity of DWT hardware is another important design aspect. An implementation with diverse hardware requirements like multipliers, buffers etc will have a lower throughput due to increased processing time and is less favorable for Polymorphic architecture.

In this article a binary coefficients 9/7 filter is implemented to allow cheaper implementation cost, higher throughput and "on-the-fly" switching to 5/3 filter architecture. The term "binary coefficients filter" refers to a filter whose coefficients can be exactly written in the form $\frac{p}{2^q}$ where p and q are integers. Thus, we have the desired rational properties of Le Gall's 5/3 filter and image compression performance similar to Daubechies' 9/7 filter.

2.1. Daubechies 9/7-Tap Bi-Orthogonal Filter

The bi-orthogonal Daubechies 9/7 filter is the most widely used filter for DWT operation. These wavelets have symmetric scaling and wavelet functions, that is, both the low-pass and high-pass filters are symmetric. This filter has excellent image compression capabilities. There are four filters that comprise the two-channel bi-orthogonal wavelet system. The analysis and synthesis low-pass filters are denoted by H_0 and G_0 respectively. The analysis and synthesis high-pass filters are denoted by H_1 and G_1, respectively, and are obtained by quadrature mirroring the low-pass filters.

$$H_1(z) = z^{-1}G_0(-z),\ G_1(z) = zH_0(-z) \tag{1}$$

Table I. Coefficients for the CDF 9/7 Filter

i	$h_0(i)$	$h_1(i)$	$g_0(i)$	$g_1(i)$
±4	0.026748757411	0	0	0.026748757411
±3	−0.016864118443	0.091271763114	−0.091271763114	0.016864118443
±2	−0.078223266529	−0.057543526229	−0.057543526229	−0.078223266529
±1	0.266864118443	−0.591271763114	0.591271763114	−0.266864118443
0	0.602949018236	1.11508705	1.11508705	0.602949018236

Table II. Coefficients for Le Gall 5/3 Filter

i	$h_0(i)$	$h_1(i)$	$g_0(i)$	$g_1(i)$
±2	−1/8	0	0	−1/8
±1	2/8	−1/2	1/2	−2/8
0	6/8	1	1	6/8

If we define $D(z) = G_0(z)H_0(z)$ the Perfect Reconstruction (PR) condition simplifies to the following:

$$D(z) + D(-z) = 2. \tag{2}$$

This equation is solved using Lagrange Half Band Filters (LHBF), $L_k(z)$ where :

$$D(z) \;=\; L_k(z) = z^k \left(\frac{1+z^{-1}}{2}\right)^{2k} \alpha(k),$$

$$\alpha(k) \;=\; \sum_{n=0}^{k-1} \left({}^{k+n-1}C_n\right) \left(\frac{2-(z+z^{-1})}{4}\right)^n. \tag{3}$$

This is simplified for $k = 4$ to get the famous Cohen-Daubechies-Feauveau (CDF) or simply Daubechies bi-orthogonal 9/7 filter. The filter coefficients are irrational and their approximate values are given in Table I. Ansari et al. [1991] discuss the derivation in detailed steps.

2.2. Le Gall's 5/3 Filter

Gall and Tabatabai [1988] solved the PR condition by substituting $D(z) = a_0 + a_2z^{-2} + a_3z^{-3} + a_2z^{-4} + a_0z^{-6}$ with the condition $a_0 \in [-\frac{1}{8}, 0]$. For $a = \frac{1}{16}$ the simplification leads to the famous Le Gall's 5/3 filter pair. The coefficients for this filter are given in Table II. This filter has lower latency than the ones studied earlier but provides lesser image compression capabilities.

$$low_{53}(i) \;=\; \frac{3}{4} \times x(i) + \frac{1}{4} \times (x(i-1) + x(i+1)) - \frac{1}{8} \times (x(i-2) + x(i+2)) \tag{4}$$

$$high_{53}(i) \;=\; x(i) - \frac{1}{2} \times (x(i-1) + x(i+1)) \tag{5}$$

3. BACKGROUND AND RELATED WORK

Our Poly-DWT architecture must enable dynamic control of the allocated resources in order to yield high performance subject to many external parameters. Although this architecture serves different needs depending on the target multimedia application, one constant across many variations is the use of wavelets for high-quality compression of image or video data.

Recent works in partial reconfiguration of FPGAs provide an insight into the state-of-the-art. Claus et al. [2008a] gives a comparison of embedded reconfigurable video-processing architectures. They propose a hybrid of two hardware platforms: one providing easy reconfiguration of modules and the other providing easy implementation with higher clock frequency, to achieve an optimal FPGA-based dynamically and partially reconfigurable platform for real-time video and image processing. The

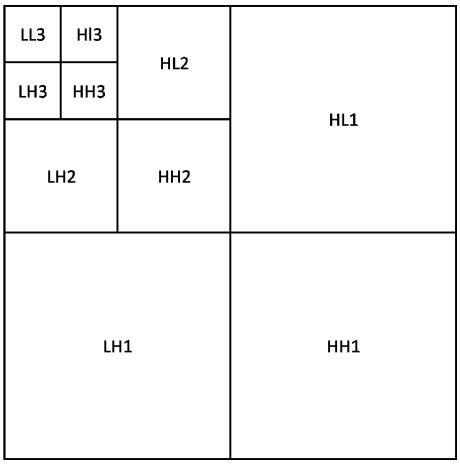


Fig. 4. Result of three level 2-D wavelet transform operation on an image.

tool ReCoBus-Builder [Koch et al. 2008] simplifies the generation of dynamically reconfigurable systems to almost a push button process. The work also describes a communication infrastructure for dynamically reconfigurable systems. Claus et al. [2008b] presents an IP core that enables fast on-chip dynamic partial reconfiguration close to the maximum achievable speed. [Paulsson et al. 2008] present a scheme for self-optimization of power and performance according to the run-time specific requirements. The work discusses power optimization of signal routing for application-specific dynamic performance requirements.

Contrary to the previously mentioned approaches, in this article we refer to reconfiguration as the dynamic switching of hardware architectures to save power resources. Thus, this switching can be implemented in both FPGA-based and ASIC-based designs. We next discuss the existing work and developments in the theory of wavelet transform and presents the motivation for hardware implementation of this algorithm. Section 3.1 discusses the development of the theory of wavelet transform, and its efficient image-processing capabilities. Section 3.2 describes some recent attempts at implementing DWT on reconfigurable platforms.

3.1. Wavelet Transform Background

The efficient representation of time-frequency information by the wavelet transform has lead to its popularity for signal processing applications. DWT provides superior rate-distortion and subjective image quality performance over existing standards. Applying a 2-D DWT to an image of resolution $M \times N$ results in four images of dimensions $\frac{M}{2} \times \frac{N}{2}$: three are detailed images along the horizontal (LH), vertical (HL) and diagonal (HH), and one is coarse approximation (LL) of the original image. LL represents the low-frequency component of the image, while LH, HL, and HH represent the high-frequency components. This LL image can be further decomposed by DWT operation. Three levels of such transforms are applied and shown in Figure 4. The coarse information is preserved in the LL3 image and this operation forms the basis of Multi-Resolution Analysis for DWT [Vetterli and Kovačevic 1995].

Spectral factorization in the frequency domain and lifting schemes are the two common schemes for achieving wavelet decomposition. The spectral factorization method first pre-assigns a number of Vanishing Moments on the Bi-orthogonal Wavelet Filter Banks (BWFBs), then obtains a trigonometric polynomial (known commonly as

a Lagrange Half-Band Filter or LHBF) and then the filter coefficients are determined according to the perfect reconstruction condition. As will be seen in the following section, we implement a spectral factorization based approach that also obtains a low-hardware implementation like that achieved from lifting by using a folding scheme.

BWFBs are commonly used for image processing but they have irrational coefficients, the associated DWT requires a high precision implementation, leading to an increased computational complexity. In a hardware implementation, rational binary coefficients can help in achieving a multiplier-free implementation of filter coefficients [Martina and Masera 2007; Redmill et al. 1997]. These multiplier-free implementations involve image reconstruction quality tradeoffs. Many other researchers have also faced the problem of reducing DWT complexity [Alam et al. 2003; Ritter and Molitor 2001; Martina and Masera 2005]. What differentiates our work is that we consider applications that could make use of runtime (not one-time) hardware resource allocation. To fulfill this requirement, we design a new polymorphic architecture that can enable dynamic control over the properties of the allocated hardware resources.

3.2. Hardware Implementation of DWT

Much research has been done in the development of DWT architectures for image processing [Benkrid et al. 2001; Ritter and Molitor 2001; Kotteri et al. 2005; Martina and Masera 2007]. A good survey on architectures on DWT coding is given by Tseng et al. [2005]. This article gives insight on hardware implementations for JPEG2000 scheme which is based on DWT computations. The computational complexity analysis of JPEG2000 by Adams and Kossentini [2000] and Lian et al. [2003] explains that EBCOT coding and DWT operations together contribute more than 80% of the overall complexity. More details of the JPEG2000 standard are given in Christopoulos et al. [2000] and Skodras et al. [2001].

The DWT architectures can be broadly classified into lifting-based, convolution-based and B-spline-based architectures. The lifting-based architectures are popular and became the mainstream because they need fewer multipliers and adders and have a regular structure. Similarly, B-spline-based architectures have been proposed to minimize the number of multipliers by using B-spline factorization [Huang et al. 2003]. However, the lifting-based architecture has a larger critical path. Convolution-based approaches have a lower critical path but require a larger number of multipliers.

In this article, we rationalize the filter coefficients that overrides the past limitations of convolution-based approaches. We introduce a multiplier-free implementation and further introduce a switching structure that enables efficient hardware resource sharing between low- and high-pass filters of DWT. By pipelining, we are able to achieve a good performance with our approach.

Chang and Hauck [2004] propose several optimization techniques aimed at providing the developer with more control over the area-to-error trade-off during data path precision optimization that would not be available with simple truncation. An error model is developed for adder and multiplier circuits. However, one of the problems faced is the uncertainty in actual error of the system that depends on the actual value of the input. The upper bound on error skews toward larger positive values as we reduce the bit allocated per pixel. In this article we make use of a dynamically reconfigurable architecture to modify the resource allocation for the system based on the image quality required by the application. Benkrid et al. [2001] discuss that the overall performance and area depends significantly on the precision of intermediate bits used in the design. This motivates us to further look at bit allocation as another aspect of polymorphism in our Poly-DWT structure.

Martina and Masera [2007] propose a multiplier-free VLSI architecture for the famous 9/7 wavelet filters. The novelty of their architecture is the possibility of combining

the 5/3 wavelet data path into the 9/7 data path, resulting in a reduced number of adders compared to other solutions. This implementation approximates the filter coefficients into two decimal places of accuracy and then implements a folded structure for achieving a hardware-efficient DWT implementation. This implementation requires 19 adders, an improvement over 21 adders required in their previous implementation [Martina and Masera 2005]. Our work obtains a different expression for wavelet filter coefficients to obtain all binary rational coefficients. This reduces the number of adders required by our implementation significantly and also achieves significantly better image reconstruction results over the original filter. As will be described in Section 4, our folded implementation reduces the number of adders to just 9.

Tay [2000] presents a technique to rationalize the coefficients of wavelet filters that will preserve bi-orthogonality and perfect reconstruction. This approach also preserves regularity of the structure by preserving most of the zeros at $z = -1$. This approach has been developed further in this article to facilitate the development of a polymorphic structure.

4. POLY-DWT FILTER

A look at Table I explains the inherent difficulties in the hardware implementation of the original Daubechies 9/7 filter. While this filter has high compression performance, it will lead to lossy compression due to truncation involved in filter coefficients in a reasonable fixed point hardware representation such as a 16-bit representation (12-bits for integer and 4-bits for fractional part values). The number of bits required for accurate representation increases as we increase the number of levels of wavelet decomposition. On the other hand, floating point implementation implies a higher hardware cost (32 bits for single precision representation). Moreover, hardware multipliers would be needed to implement this in our design with reasonable precision.

We alleviate this problem by searching for an integer coefficients filter that can offer a higher PSNR at a smaller word size. A parameterized filter design allows us to obtain a family of 9/7 filters. This new design is then searched for rational coefficients to obtain new filters to alleviate the previously mentioned problems.

4.1. Parameterized Filter Design

A parameterized design alleviates the problem of irrational coefficients. Tay [2000] poses this constraint on $D(z)$ to derive the binary rational coefficients and achieve new sets of 9/7 filters by adding more degrees of freedom to the original LHBF equation (by introducing a free parameter α):

$$H_0(Z) = K_h(Z+1)(Z^3 + AZ^2 + VZ + C) \tag{6}$$

$$G_0(Z) = K_g(Z+1)^2(Z+\alpha) \tag{7}$$

$$D(Z) = K_h K_g(Z+1)^3(Z+\alpha) \times (Z^3 + AZ^2 + BZ + C). \tag{8}$$

The PR condition on $D(Z)$ gives simultaneous constraint equations which simplify to give solutions for A, B, and C (and simultaneously for the filter coefficients) in terms of α:

$$A = -(3+\alpha) \tag{9}$$

$$B = \frac{9\alpha^3 + 35\alpha^2 + 48\alpha + 24}{3\alpha^2 + 9\alpha + 8} \tag{10}$$

$$C = \frac{8(1+\alpha)^3}{3\alpha^2 + 9\alpha + 8}. \tag{11}$$

Here, we have $Z = (z+z^{-1})/2$. Setting α to -1.6848 gives back the original 9/7 filter pair.

Table III. Analysis High-Pass Filter Coefficients (H_1) for the Bi-Orthogonal 9/7 Tap Filter

$i\backslash\alpha$	1.6848	-1.667	-1.8	-2
±3	0.091271763114	1/16	1/16	1/16
±2	-0.057543526229	$-1/16$	$-1/16$	0
±1	-0.591271763114	$-9/16$	$-9/16$	$-9/16$
0	1.11508705	9/8	9/8	1

Table IV. Analysis Low-Pass Filter (H_0) Coefficients for the Bi-Orthogonal 9/7 Tap Filter

$i\backslash\alpha$	1.6848	-1.667	-1.8	2
±4	0.026748757411	1/32	1/32	1/64
±3	-0.016864118443	$-1/32$	0	0
±2	-0.078223266529	$-1/16$	$-3/32$	$-1/8$
±1	0.266864118443	9/32	1/4	1/4
0	0.602949018236	19/32	5/8	23/32

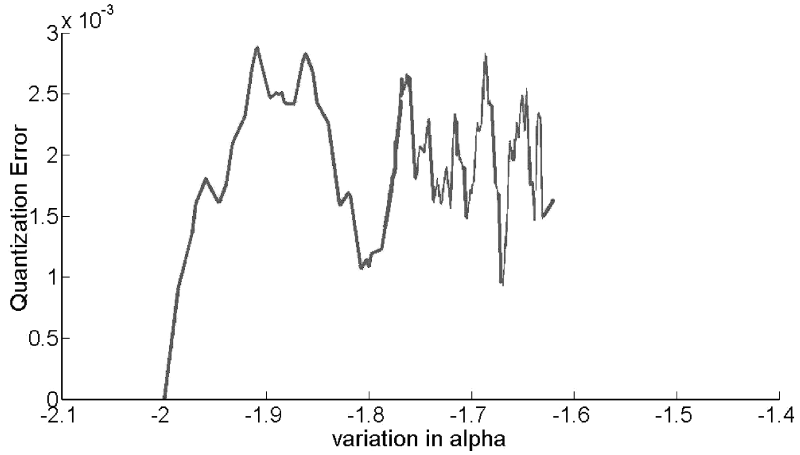


Fig. 5. Numerical Analysis of Quantization error for seven bit finite representation of filter coefficients.

4.2. Numerical Study

The parameter α can be varied to achieve a family of bi-orthogonal filter pairs for DWT implementation. Setting $\alpha = -1.6848$ gives us the CDF-9/7 filter that have been proven to have good compression performance. Next, we perform a numerical study to explore a set of binary coefficients filter which is in close proximity to the CDF-9/7 filter. We ran MATLAB experiments to obtain the quantization error for the filter coefficients with α varying from -1.6 to -2 (in vicinity of the $\alpha = -1.6848$ value). The results are presented in Figure 5. It can be observed that the minimization of this error occurs at $\alpha = -2$, where quantization error drops down to 0. A zero quantization error indicates that the filter coefficients derived with $\alpha = -2$ are (exactly) rational. We can also observe local minima of curves around two regions in the vicinity of $\alpha = -1.6848$ (at $\alpha = -1.66$ and $\alpha = -1.8$ approximately). We also derive approximate filter coefficients from these minima to obtain a binary coefficients 9/7 filter. These filter coefficients are reported in Tables III and IV.

4.3. Candidate Filters

Let us consider an input signal $x(i)$. The low- and high-pass outputs of this filter ($low(i)$ and $high(i)$, respectively) are obtained by convolution of $x(i)$ with $h_0(i)$ and $h_1(i)$,

respectively:

$$low(i) = \sum_{k=-4}^{k=4} h_0(k) \cdot x(i-k), \tag{12}$$

$$high(i) = \sum_{k=-3}^{k=3} h_1(k) \cdot x(i-k). \tag{13}$$

Substituting the values of filter coefficients from Tables II, III, and, IV, we can factorize our 9/7 filter coefficients in terms of 5/3 filter outputs. The subscripts A, B, and C are used to denote the filters obtained with $\alpha = -1.67$, -1.8, and -2, respectively:

$$\begin{aligned} low_A(i) \;=\; & \frac{19}{32} \times x(i) + \frac{9}{32} \times (x(i-1) + x(i+1)) - \frac{1}{16} \times (x(i-2) + x(i+2)) \\ & - \frac{1}{32} \times (x(i-3) + x(i+3)) + \frac{1}{32} \times (x(i-4) + x(i+4)) \end{aligned} \tag{14}$$

$$\begin{aligned} high_A(i) \;=\; & \frac{9}{8} \times x(i) - \frac{9}{16} \times (x(i-1) + x(i+1)) - \frac{1}{16} \times (x(i-2) + x(i+2)) \\ & + \frac{1}{16} \times (x(i-3) + x(i+3)) \end{aligned} \tag{15}$$

$$\begin{aligned} low_B(i) \;=\; & \frac{5}{8} \times x(i) + \frac{1}{4} \times (x(i-1) + x(i+1)) - \frac{3}{32} \times (x(i-2) + x(i+2)) \\ & + \frac{1}{32} \times (x(i-4) + x(i+4)) \end{aligned} \tag{16}$$

$$\begin{aligned} high_B(i) \;=\; & \frac{9}{8} \times x(i) - \frac{9}{16} \times (x(i-1) + x(i+1)) - \frac{1}{16} \times (x(i-2) + x(i+2)) \\ & + \frac{1}{16} \times (x(i-3) + x(i+3)) \end{aligned} \tag{17}$$

$$\begin{aligned} low_C(i) \;=\; & \frac{23}{32} \times x(i) + \frac{1}{4} \times (x(i-1) + x(i+1)) - \frac{1}{8} \times (x(i-2) + x(i+2)) \\ & + \frac{1}{64} \times (x(i-4) + x(i+4)) \end{aligned} \tag{18}$$

$$high_C(i) = x(i) - \frac{9}{16} \times (x(i-1) + x(i+1)) + \frac{1}{16} \times (x(i-3) + x(i+3)) \tag{19}$$

The original Daubechies 9/7 filter has $\alpha = -1.68$. Thus, the compression performance of A will be slightly greater than B and C. However, we can also see that the C architecture requires fewer number of addition operation. The simpler coefficients value in C (coefficients being 0 or easily represented in exponents of (2) promises a cheaper hardware implementation. This implies a tradeoff between image reconstruction quality vs. hardware resources required by various filters. In the next section, we discuss the hardware resource requirements of these architectures.

4.4. Hardware Architectures

We performed several optimization steps to reduce the cost of underlying hardware. The following optimization steps were performed:

—Observe in Tables II, III, and IV that the coefficients of $x(i \pm k)$ are the same. Thus, they can be grouped together to reduce the hardware complexity.

$$w_0 = x(0), \tag{20}$$
$$w_1 = x(i-1) + x(i+1), \tag{21}$$
$$w_2 = x(i-2) + x(i+2), \tag{22}$$
$$w_3 = x(i-3) + x(i+3). \tag{23}$$

The Daubechies 9/7 filter requires 9 data values - four each corresponding to four previous and next values and one for the present pixel value. After this optimization, we reduced this number from nine to five. This also reduces the requirement of multipliers in implementing equations such as Eq. (12) and (13) in hardware from nine to five.

—Division by binary coefficients (e.g., 1/64, 1/16, 1/4) was performed using arithmetic shift operations. This eliminates the need for multipliers in the circuits. The coefficients as given in Tables III and IV are rational and most of them have some simple binary value. Therefore, we switch our design to a multiplier-free design requiring limited adders in the implementation.

—The input stream was pipelined. Thus, as shown in Figure 6 our architecture takes one pixel (or channel input) as the input and outputs the low- and high-pass signal coefficients with a finite latency. This helps us to achieve a good throughput and a higher clock frequency. The pipeline stages are implemented by clocking the cascaded registers to the left in the figure.

Figure 6(a-c) provides a visual overview of the three designs with the value of $\alpha = -1.67, -1.8$, and -2, respectively. As can be seen in Figure 6, our Le Gall's 5/3 filter implementation requires only six adder/subtracter units. Our 9/7 filter implementations for $\alpha = -1.67$ required 19 adders. For $\alpha = -1.8$, our design requires 17 adder/subtracter units. But we observe that the design for $\alpha = -2$ requires only 12 adder/subtracter units. This is a significant improvement over any reported existing work as reported in the experiment section.

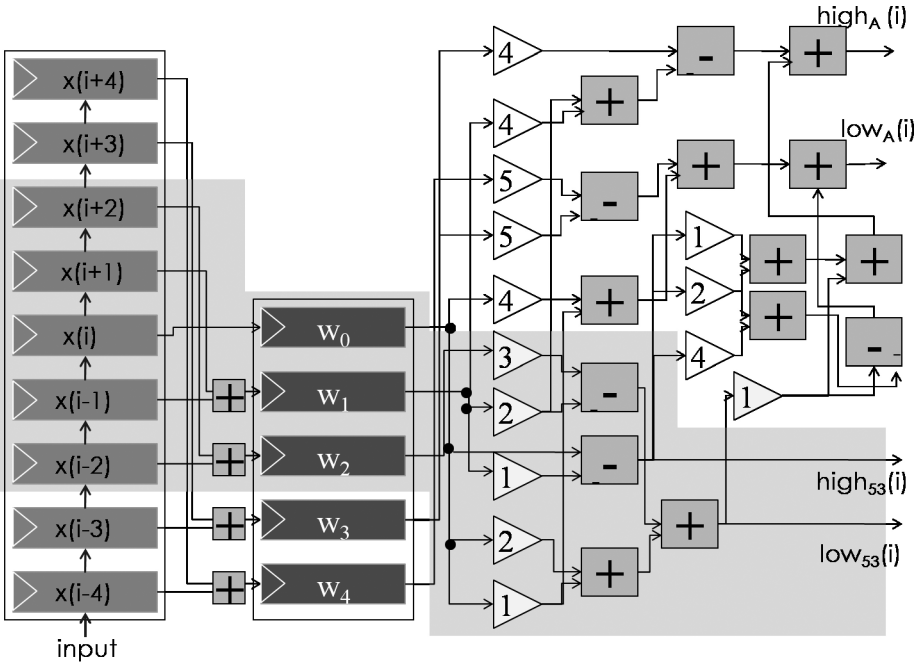
As described in Figure 1, the reconfigurable implementation must allow dynamic switching between wavelet filters. Our implementation allows for easy enabling and disabling of the extra hardware to obtain the choice between a more power-efficient binary 5/3 filter versus a more compression-efficient 9/7 filter. In the remainder of this section we describe an architecture to allow for this dynamic switching. Let us consider an input signal $x(i)$. The low- and high-pass outputs of this filter ($low(i)$ and $high(i)$, respectively) are obtained by convolution of $x(i)$ with $h_0(i)$ and $h_1(i)$, respectively:

$$low(i) = \sum_{k=-4}^{k=4} h_0(k) \cdot x(i-k), \tag{24}$$

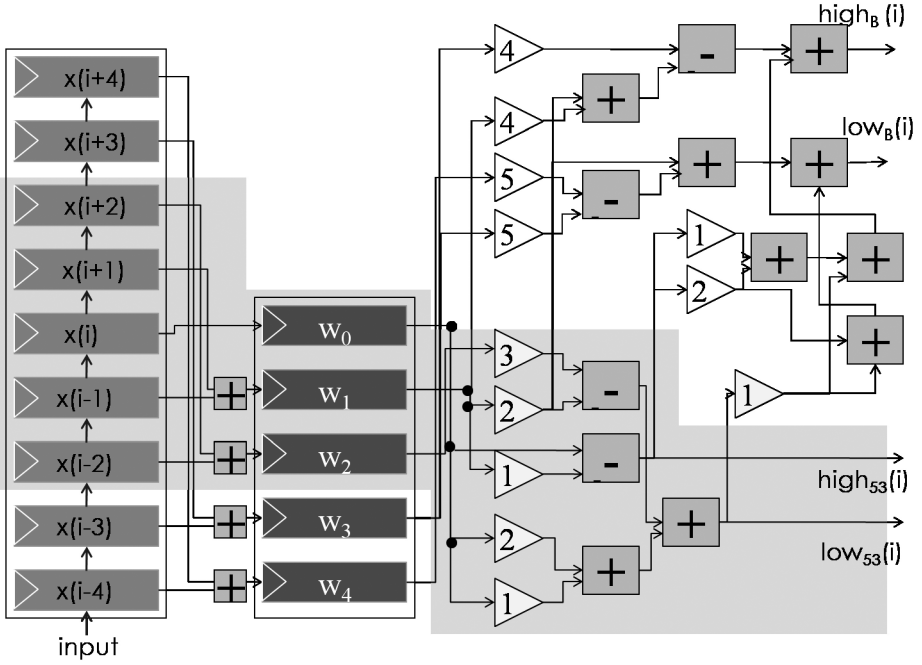
$$high(i) = \sum_{k=-3}^{k=3} h_1(k) \cdot x(i-k). \tag{25}$$

Substituting the values of filter coefficients from Table II, III, and IV, we can factorize our 9/7 filter coefficients in terms of 5/3 filter outputs.

$$\begin{aligned} low_A(i) &= 1/2 \times low_{53}(i) - (1/4 + 1/16) \times high_{53}(i) \\ &\quad +(1/2 + 1/32) \times w_0 + 1/32 \times (w_4 - w_3) \end{aligned} \tag{26}$$
$$\begin{aligned} high_A(i) &= 1/2 \times low_{53}(i) + (1/2 + 1/4) \times high_{53}(i) \\ &\quad -1/4 + 1/16) \times w_1 + 1/16 \times (w_3) \end{aligned} \tag{27}$$

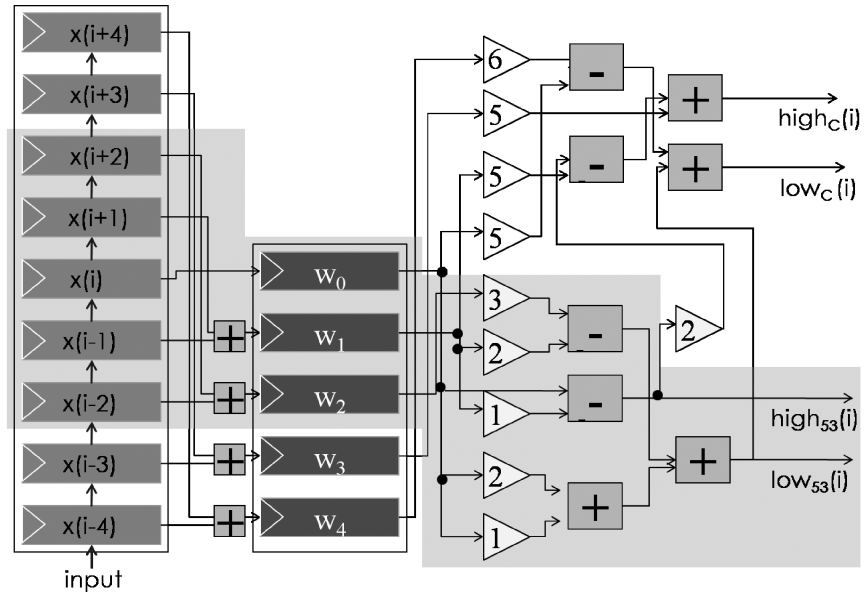


(a) $\alpha = -1.67$

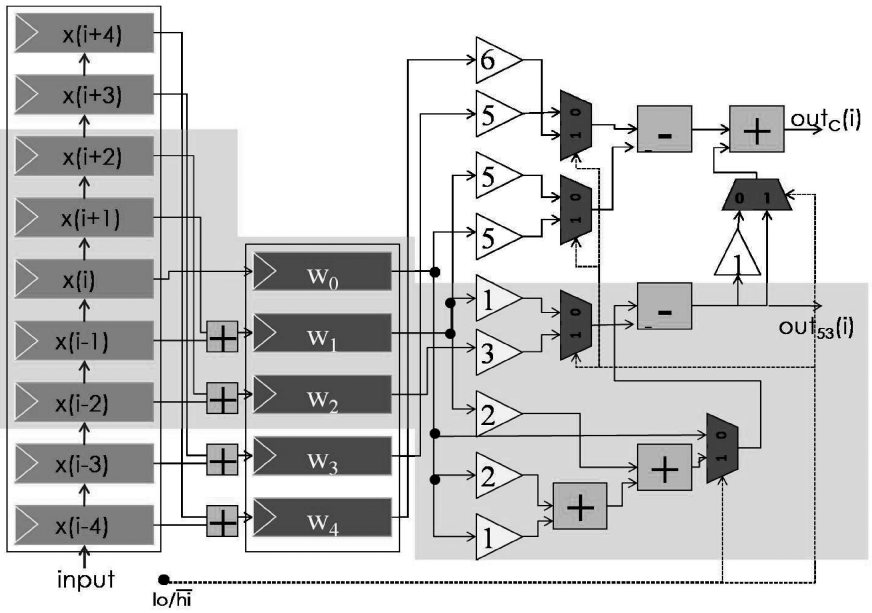


(b) $\alpha = -1.8$

Fig. 6. Continued.



(c) $\alpha = -2$



(d) $\alpha = -2$ (folded)

Fig. 6. Hardware architectures for bi-orthogonal 9/7 filter

$$\begin{aligned} low_B(i) &= 1/2 \times low_{53}(i) + 1/4 \times high_{53}(i) \\ &\quad +1/32 \times (w_4 - w_3) \qquad (28) \\ high_B(i) &= 1/2 \times low_{53}(i) + (1/2 + 1/4) \times high_{53}(i) \\ &\quad -(1/4 + 1/16) \times w_1 + 1/16 \times (w_3) \qquad (29) \end{aligned}$$

$$low_C(i) = low_{53}(i) - 1/32 \times w_0 + 1/64 \times w_4 \quad (30)$$

$$high_C(i) = 1/2 \times high_{53}(i) - 1/32 \times w_1 + 1/32 \times w_3 \quad (31)$$

Figure 6(a-c) provides the implementation details of these architectures. The dark (yellow) region is the hardware required for the implementation of Le Gall's 5/3 filter. The architecture has registers, adders, and multiplexers. The right shift operation (can be implemented by adjusting the wires) is represented by small triangles. A triangle with the number "x" means a shift to the right over "x" positions, or a division by 2^x. All the architectures are designed as extensions of Le Gall's 5/3 filter. This gives the feature of 'on the fly' switching from 9/7 filter to Le Gall's mode of operation.

The low- and high-pass filter outputs can be $low_{A/B/C}(i)$ and $high_{A/B/C}(i)$, or $low_{53}(i)$ and $high_{53}(i)$ depending upon the mode of operation. When operating in 5/3 filter mode only the yellow shaded region of the architecture would be used thus reducing considerably the power consumption of the system. This figure shows the conceptual design and architecture and does not include the pipeline stages of these structures. A folded architecture can be developed for the $\alpha = -2$ case where the low- and high-pass output coefficients are dependent only on low- and high-pass values respectively of 5/3 filter. This is presented in Figure 6(d). This design requires only nine adders in the circuit.

5. FIXED POINT IMPLEMENTATION

An image channel is generally represented at 8-bit precision. This encourages us to develop a fixed-point hardware. We avoid the floating point implementation of the system to avoid nonoptimal usage of resources. Chang and Hauck [2004] discuss the issues involved in the fixed, point analysis with respect to the output error. There are two conflicting issues that affect the decision to decide the hardware bit allocation for internal representation of variables:

(1) Increased number of bits generally implies better performance in terms of image quality and reduced error.
(2) Reduced number of bits imply a better hardware utilization, and lower power consumption.

For certain applications such as a static HDTV encoding system, we may always require a large number of bits that ensure high-quality and high-resolution multimedia transmission. However, certain applications such as remote tele-medicine applications and remote distributed surveillance applications are highly power and performance sensitive. They may require a dynamic tradeoff. The Poly-DWT provides a good trade-off in achieving a dynamic hardware reconfiguration for such applications. Similarly, Chang and Hauck [2004] report that the error (in image reconstruction in case of DWT) is skewed or biased, only in the positive direction. Thus, static analysis may not be applicable in all situations and we need custom hardware to adapt itself according to the present conditions. The image statistics (like Peak Signal to Noise Ratio (PSNR), Mean Absolute Deviation (MAD)) provide the system a performance feedback and allows it to take steps to lead to a more efficient representation. These metrics can be used as performance measure of image compression systems and we can switch hardware to reach a desired compression level with minimum hardware resources.

We present a simple scheme to change the bit allocation for hardware implementation. The main factors or sources for the change in hardware bit allocation can be summarized in the following headings.

—*Functional Requirements of the Chip.* There may be several computational kernels such as image enhancement, noise filtering, etc, which may be optionally required for a multimedia application. Depending on input from the source, some of them may not be required to be functional at all times. The extra hardware available in such cases can be dedicated to the Poly-DWT architecture to improve its performance.
—*Quality Requirements of the Application.* Many DWT kernels or instances of DWT hardware may be required by different applications. Moreover, with the change in input images we may dynamically require different levels of accuracy.
—*Level of Decomposition using DWT.* In image compression algorithms such as SPIHT, CEZW, and EBCOT more than one level of DWT operation is done. Fry and Hauck [2005] discuss the changes in numeric range in higher level decomposition using DWT. For example, the eight bit input can have maximum magnitude of 255 and can be well represented using 8.0 fixed point format representation (8 bits to represent integer and 0 bits for fractional part). An analysis of the coefficients of each filter bank shows that a 2D low-pass FIR filter at most increases the range of possible numbers by a factor of 2.9054. As a result, the coefficients at different wavelet levels require a variable number of bits above the decimal point to cover their possible ranges. At fourth wavelet decomposition level, 17-bit representation may be required to accommodate the magnitude range of coefficients. A dynamic word width allocation may make a lower-level DWT kernel fit for decomposition at higher level if required by the application.
—*User Preferences.* In our proposed system, the user has the final say in all the subjective image quality/cost trade-offs. Applications and users may differ in their subjective view of good performance of the system. Chang and Hauck [2004] also discuss the importance of defining a user-defined error constraint.
—*Other Considerations.* A Poly-DWT implementation may include other considerations like the number of DWT kernels required, separation/folding of row and column processing DWT kernels etc. We have not discussed these aspects in our present Poly-DWT analysis and they are left as a future work.

6. HARDWARE (RE)-ALLOCATION

Poly-DWT allows several levels of hardware resource (re-)allocation to obtain a power-efficient design, which are explained as follows.

(1) The number of DWT kernels in the wavelet decomposition can be varied depending upon the application requirements.
(2) On-the-fly switching of filter design from 9/7 to 5/3 filter architecture in finite cycles latency.
(3) The number of bits allocated for internal registers in the design can be varied to a obtain an application-specific trade-off between clock frequency and reconstruction quality vs. hardware usage.

The variations in number of DWT kernels in wavelet decomposition is specific to the requirements of the multimedia encoding scheme and its dynamic requirements. In this paper, we therefore restrict our discussion to reconfiguration of design of the individual DWT kernels to meet the performance vs. power trade-off dynamically in hardware. Figure 7 gives the architecture design of poly-DWT kernel to achieve these tradeoffs.

6.1. "On-the-fly" Switching

We first consider "on-the-fly" switching of filter designs from 9/7 to 5/3 architectures. The $switch_{hw}$ signal in Figure 7 is used to switch between 5/3 and 9/7 architectures. The two multiplexers (unshaded in Figure 7) ensure the correctness of the input and

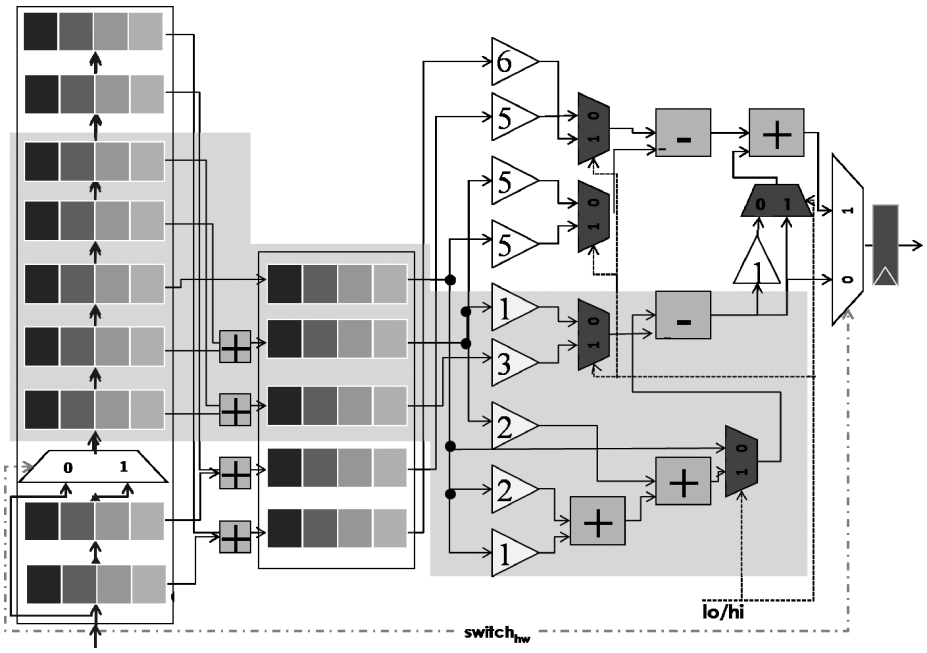


Fig. 7. Architectural details of poly-DWT to facilitate "Reconfiguration".

output of poly-DWT hardware. As seen in this figure, we can divide the hardware into two categories.

(1) *Type A Hardware.* The hardware common to both 5/3 and 9/7 filter architectures is called as type A hardware and it is unaffected by "on-the-fly" switching. This includes the registers and adders in the shaded portion of design in Figure 7.
(2) *Type B Hardware.* The hardware used by 9/7 filter architecture, which is obsolete to 5/3 filter, is called as type B hardware. This hardware is switched off when $switch_{hw}$ signal is changed to 0. This corresponds to the registers and adders in the unshaded portion of design in Figure 7.

The following steps are involved in switching from 9/7 to 5/3 filters (the 5/3 filter hardware is shaded in Figure 6).

(1) The input pipeline for the 5/3 filter is smaller than the 9/7 filter. In order to use the same pipeline we need a latency of two cycles to ensure that the pipelining registers have proper inputs. The values in the pipeline registers ($x(i-4)$ and $x(i-3)$) are pipelined to the 5/3 filter hardware before they are switched off.
(2) The extra hardware for computation can be switched off in a single clock cycle. This can be enforced by driving the signal $switch_{hw}$ from 1 to 0 (shown in Figure 8(b) and explained in next section);
(3) The input and output multiplexer can be switched from input port 1 to input port 0 in one cycle.

Since the previously mentioned operations can be performed together, we require only a latency of two cycles to switch from a 9/7 to a 5/3 filter. A similar argument can be constructed to explain that it would take a latency of two cycles to switch from 5/3 to 9/7 filter architecture.

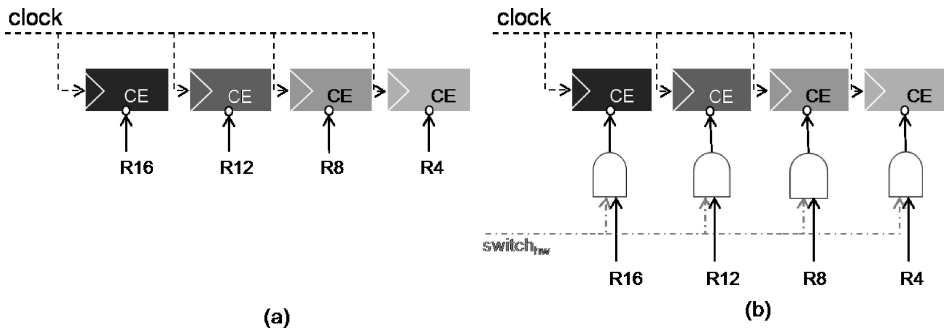


Fig. 8. Register level details to enable reconfiguration (a) Type A architecture and (b) Type B architecture

While the proposed architecture is capable of switching between 5/3 and 9/7 filter architectures at runtime of a few *ns*, such a design will incur a large overhead in transmitting control information to ensure the correctness of the output at the decoder (we will need to send 1 bit per clock cycle for one filter kernel used). However, in practical scenarios, we can restrict the switching between 5/3 and 9/7 filters between different levels of wavelet decomposition. Thus, the overhead involved in such a switching is reduced to a few bits (3-10 bits per frame) and can be integrated into frame header.

6.2. "Bit-width" switching

We discuss the scheme for bit-width switching in this section. We break the internal registers in design into multiples of four bit registers. Thus, a $N = 16$ bits register is represented as four 4-bit registers. As shown in Figure 7, the registers are represented as four 4-bit registers. Figure 8 explains the working of "bit-width" switching scheme with individual registers. The four signals $R4$, $R8$, $R12$, and $R16$ are used to switch the registers on or off at runtime. When $R4$, $R8$, and $R12$ are on, the register has 12 bits available for use while the other 4-bit register is switched off to save power. This is done with the help of chip enable (CE) signal as indicated in the Figure 8. Similar changes can be made to the design of adders to partially switch off the LUTs corresponding to an adder hardware. Figure 8(a) explains the bit-width switching of type A hardware. The two inputs $switch_{hw}$ and the register select input ($R4/R8/R12/R16$) are ANDed to get the chip-enable (CE) signal for individual 4-bit type B registers. This enabling/disabling of registers for type B hardware is illustrated in Figure 8(b).

The dynamic power consumption of a circuit is given by the following equation:

$$P = ACV^2F,$$

where A is the activity factor ($0 \leq A \leq 1$), C is the switched capacitance of the circuit, V is the supply voltage and F is the clock frequency. By switching off the extra hardware, we reduce the switched capacitance C of the circuit, thereby obtaining a useful dynamic tradeoff between the power and performance constraints.

7. EXPERIMENTS

This section presents quantitative results for the performance of Poly-DWT architecture presented in this article. We evaluate our approach on the Xilinx Virtex-V XC5VLX30 FPGA by generating the different DWT architectures. The polymorphic architecture presented in this article has been analyzed in terms of image reconstruction and kernel area considerations. As previously mentioned, the tradeoff between the two is dynamically reached in a polymorphic architecture.

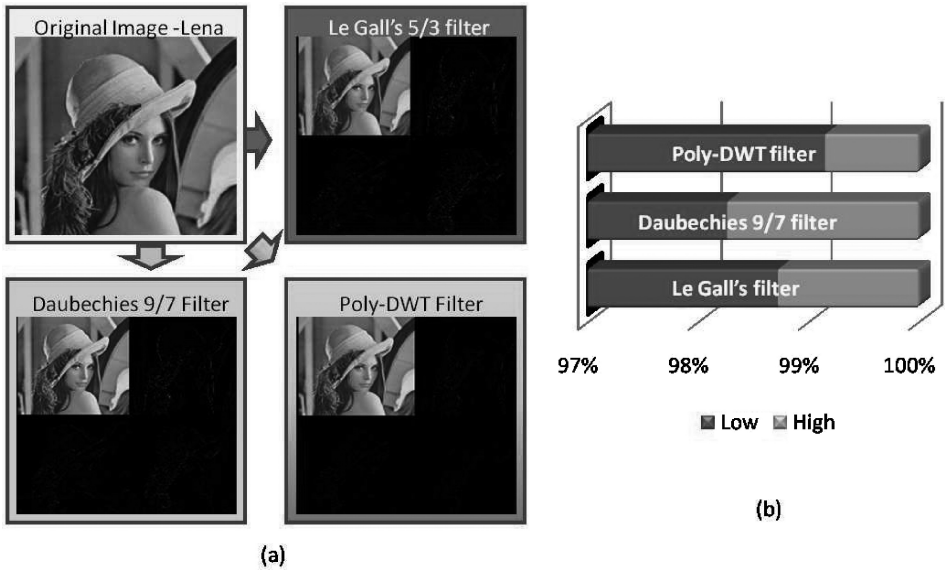


Fig. 9. (a) Results of one level of DWT and (b) Energy decomposition by respective filters.

We present the results of analysis for various word widths for internal and external configurations of DWT kernel and also examine the performance of different kernels. The standard color test images (e.g., Lena, Barbara) were used for the purpose of simulation. Each pixel in a color image has 3 channels, with 8 bits of data per channel. Unless otherwise specified, we used 8.4 fixed point arithmetic for internal computations.

Our design is written in VHDL and synthesized using Xilinx ISE 9.1i. ModelSim simulations were performed to test the waveforms. The more detailed analysis of image reconstruction performance of various filters is performed in MATLAB. To verify the correctness of the various filters implemented in the FPGA, we compared it against a pure software implementation on a Intel Core 2 Duo processor running at 2.0 GHz. Both implementations generate the same numerical results for transformed output. In the following sections, we analyze the working of our proposed DWT hardware with respect to area, performance and quality perspectives.

7.1. Image Reconstruction Quality

The proposed Poly-DWT filter gives a more efficient representation than the original Daubechies 9/7 filter as well as the Le Gall's 5/3 filter as illustrated in Figure 9(a). It can be seen in the figure that Poly-DWT provides provides very little high-pass information (white marks in black background in higher frequency subbands). The reduction in high-level information in our Poly-DWT filter makes it more suitable for the compression applications.

A more accurate representation over fixed point hardware gives a better image reconstruction for Poly-DWT filter than the Daubechies 9/7 filter. Results over several test images showed similar results. The bars in Figure 9 illustrates the superior performance of the Poly-DWT filter for limited hardware resources. The ratio of energy of the low- and high-pass components is measured. Poly-DWT is found to outperform other filters in retaining low pass energy. This property, also known as *energy compaction property*, of the filter is helpful to achieve a better compression efficiency.

The image compression performance of Poly-DWT filter was evaluated on a SPIHT image coder [Said and Pearlman 1996]. We tested the performance on an open-source

Table V. Image Compression Performance on SPIHT Coder (PSNR Values)

Image	Bitrate=0.5 bpp			Bitrate=2 bpp		
	Daub. 9/7	Poly-DWT	Martina,07	Daub. 9/7	Poly-DWT	Martina,07
lena	28.213	29.46	27.7	38.47	38.17	36.5
surveillance	26.1	28.1	26.54	38.41	42.21	39.21
lecture	34.35	33.8	32.73	48.3	51.25	43.71
helicoptor	33.75	35.7	35.01	48.59	54.72	47.14

Table VI. Hardware Acceleration on a Virtex-5 XC5VLX30 FPGA (time in μs)

Image	Le Gall 5/3 Filter			Daubechies 9/7 Filter			Poly-DWT 9/7 Filter	
	SW	HW	Speedup	SW	HW	Speedup	HW	Speedup
CIF	1420	197	7.06×	2980	330	9.03×	288	10.35×
Q-CIF	370	68	5.45×	790	91	8.68×	77	10.26×

filter bank based implementation provided by Tian [2008]. We chose the intermediate variables in 9.4 fixed-point format for this experiment setup. In case of low bit-rate applications, this property helps in better reconstruction of images from low pass coefficients. The performance over some test sequences has been reported in Table V. The results are reported over bit rates of 0.5 bpp (bits per pixel) and 2 bpp. It can be seen that the compression efficiency of Poly-DWT filter is comparable to Daubechies 9/7 filter. A performance comparison with another multiplier-free implementation provided by Martina and Masera [2007] illustrates that our design requires a fewer number of adders and gives a higher compression performance as evident by higher PSNR values.

7.2. Hardware vs Software Performance

The hardware performance of DWT kernels proposed in the article was compared with a software based implementation on the PC platform. Table VI gives the speedup achieved by an FPGA based implementation of DWT kernels. The software implementation of both Daubechies 9/7 filter and Poly-DWT (9/7) filter takes the same time as the number of filter taps in both cases is the same. The FPGA based design outputs one pixel per clock cycle for every DWT kernel. The computation times for one level of DWT for different image sizes is presented in Table VI. The is reported in microseconds (μs). The proposed Poly-DWT filter obtains a speedup of about a factor of 10 for CIF images (standard images of size 352×288 pixels). The speedup for Q-CIF images (Quarter-CIF) is also about 10. The smaller speedup in smaller sized images is attributed to the overheads in I/O operations which are more significant in the case of small-sized images. A line-based image scan architecture [Chrysafis and Ortega 1998] is used for data I/O operations.

The results as summarized in Table VI show the advantages of a hardware implementation of this class of algorithms. This is due to the fact that the required calculations are simple, allowing for a high throughput implementation. By pipelining the individual adder and add operations, we were able to achieve very high clock frequencies (394 MHz on our target Virtex-5 platform and 4 bits word length). The actual speedup achieved by the Poly-DWT kernel (Table VII) over Daubechies filter is greater (three times more) than the results indicated in Table V because of memory access computations involved in image compression results.

7.3. Hardware Comparison

Direct implementation of the CDF-9/7 filter gave a clock frequency of 107 MHz, while requiring 9 multiplier units. A clock frequency of 110 MHz was reported when we forced the design to map the constant multiplications into Lookup Tables. [Martina and Masera 2007] implement Daubechies 9/7 filter with approximate coefficients and

Table VII. Comparison of Binary Filter Features and Hardware Resources Requirements.

Features	Daub. 9/7	$\alpha = -2$ folded	$\alpha = -2$	$\alpha = -1.8$	$\alpha = -1.67$	Tay, 2001	Kotteri, 2005	Huang, 2001	Martina, 2007	Martina, 2005
Adders	15	9	12	17	19	19	15	8	19	21
Multipliers	9	0	0	0	0	0	0	4	0	0
PSNR	A	B	B	A	A	C	C	A	B	B
Reconf.	N	Y	Y	Y	Y	N	N	N	Y	N
Registers	144	208	213	253	294	-	-	-	-	-
LUTs	80	175	194	217	289	-	-	-	-	-
Bit Slices	210	245	259	311	375	-	-	-	-	-
Clock(MHz)	107	389	317	311	310	-	-	-	200	-

report a clock frequency of about 200 MHz through a multiplier-free implementation, targeting 0.13 μm VLSI technology.

Table VII summarizes the performance of our Xilinx Virtex-V implementation, and compares our results with other recent works. All the parameterized binary implementations outperform the existing implementations in terms of number of required adders and clock frequency.

Our initial nonpipelined design obtained a clock frequency of about 108 MHz, due to its long critical path. The critical path of the circuit lies from the w_i registers to the final output $low_C(i)$ or $high_C(i)$, passing through signals $low_C(i)$ or $high_C(i)$. We then pipelined this computation into several stages and obtained a faster implementation. The $\alpha = -2$ architecture showed a clock frequency of about 317 MHz. This design requires less FPGA resources (registers and LUTs) than the $\alpha = -1.67$ and $\alpha = -1.8$ architectures and is most fit for Poly-DWT implementation.

The folded architecture variant for $\alpha = -2$ was also implemented, resulting in a faster clock frequency and less adders (leading to fewer logic slices). The design of binary coefficients filter also helped us to achieve perfect reconstruction of image signals. This proposed architecture can run (over line-based DWT architectures) at 389 MHz, enabling it to process High Definition Video frames (1440×1080) in an estimated 5 ms time. As previously mentioned, the shaded (yellow) regions in Figure 6 show the baseline 5/3 filter implementation. Thus, the architecture can be optimized to switch on-the-fly to 5/3 mode in order to save power. The folded architecture and the simple architecture of Poly-DWT filter both have the same performance in terms of image reconstruction and they differ only in hardware requirements. The input data width was 8 bits corresponding to one channel of an image stream. The proposed binary filter reaches perfect reconstruction with lesser number of bits than the Daubechies 9/7 filter. Thus, the overall area requirements are less. The hardware resources utilized in these DWT kernels are summarized in Table VII. Here, a comparison of hardware resources utilization is provided against existing works. Martina and Masera [2007] present a multiplier-free implementation which is suitable for polymorphic switching between 9/7 and 5/3 filters. However, they approximate the original Daubechies filter coefficients to two decimal places which leads to its poor PSNR performance. Our architecture provides both more efficient hardware usage and better compression performance.

Figure 10 shows the change in synthesized clock frequency for the various implementations of DWT with varying input word width. The change in external data width as shown in Figure 10 leads to reduction of clock frequency and hence reduced throughput.

Zhang et al. [2007] present a switching between 5/3 and 9/7 filters using partial reconfiguration of the bit streams and a lifting-based implementation of the DWT. They used a platform based on Xilinx Virtex-4 FPGA for experimental implementation. However, this implementation requires a switching time of 40.2 ms. Thus, this system introduces a delay/lag of 2 frames (at CCIR resolution of 720×576 pixels per frame

Table VIII. Performance Comparison using Standard Cell Libraries

	Poly-DWT	Le Gall's	Daub. 9/7	[Martina and Masera 2007]*
Area	2135	1370	6693	-
Cells	544	194	1022	-
Clock Frequency	500	500	300	200

*Martina and Masera [2007] reports a gate count of 2.68K using a 130nm cell library.

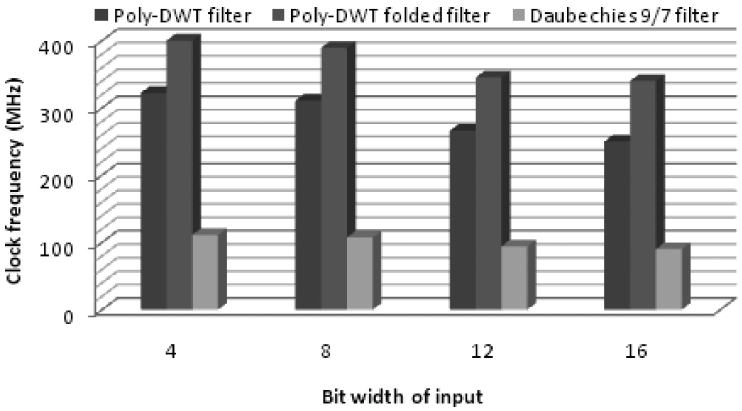


Fig. 10. Change in FPGA clock frequency(MHz) for variable word widths for various filters.

and 50 MHz clock). As compared to these results, poly-DWT has a very small switching time of two clock cycles (equivalent to 5.14 ns, assuming a 389 MHz clock).

7.3.1. ASIC Synthesis. In order to make a more fair comparison with related work, we also synthesized our Poly-DWT architecture to ASIC technology. We used the Synopsys Design Compiler environment to perform our experiments using the freePDK 45nm cell library [Stine et al. 2007]. The results of ASIC synthesis indicate that we can achieve a clock frequency comparable to Le Gall's filter with an insignificant increase in the number of cells in the design (as reported in Table VIII). We were able to achieve a clock speed of 500 MHz for the folded 9/7 filter design.

7.4. Dynamic Bit Allocation

In this section we study the effect of bit allocation on the clock frequency and image quality. The implementation used fixed point arithmetic over VHDL. First, the input data was kept at 8.0 format and the word width of internal registers was changed. Figure 11 shows the change in reconstruction quality of the images depending on changes in hardware resources (single bit registers or flip-flops). The x axis here refers to the total number of bits given to an internal processing register. Figure 12 compares the implementation of our structure with other filters. It is observed that changes in bit-width of internal registers from 9.0 to 9.6 fixed point representation leads to a linear increase in hardware requirements (number of single bit registers or flip-flops) and a slight decrease in achievable clock frequency. The folded Poly-DWT filter register usage on an FPGA chip approaches the implementation of 5/3 filter, while its compression performance approaches the Daubechies' 9/7 filter. This indicates the hardware efficient feature of our design.

7.5. Real-World Application

We consider a real-time scenario where we propose a DWT based video-surveillance system. Lake Pontchartrain Causeway in southern Louisiana has a bridge that runs

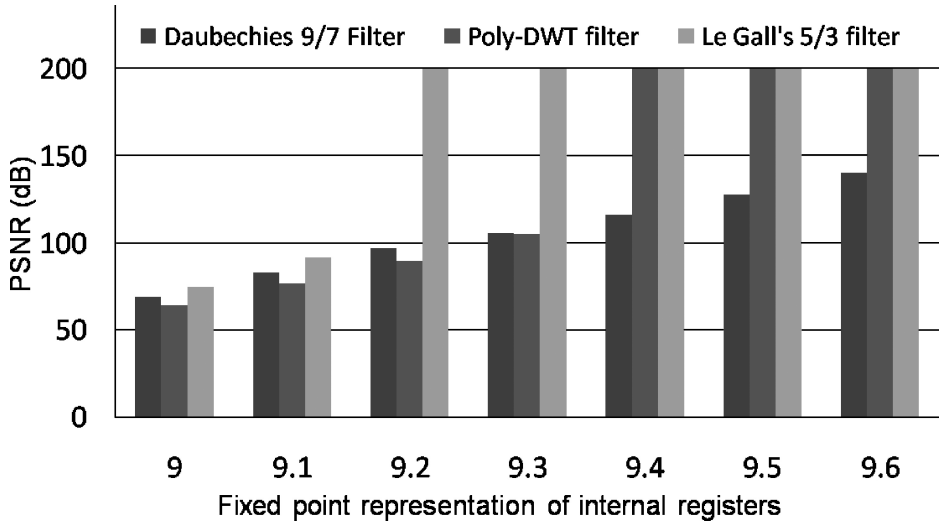


Fig. 11. Plot of PSNR vs the number of bits alloted for internal registers

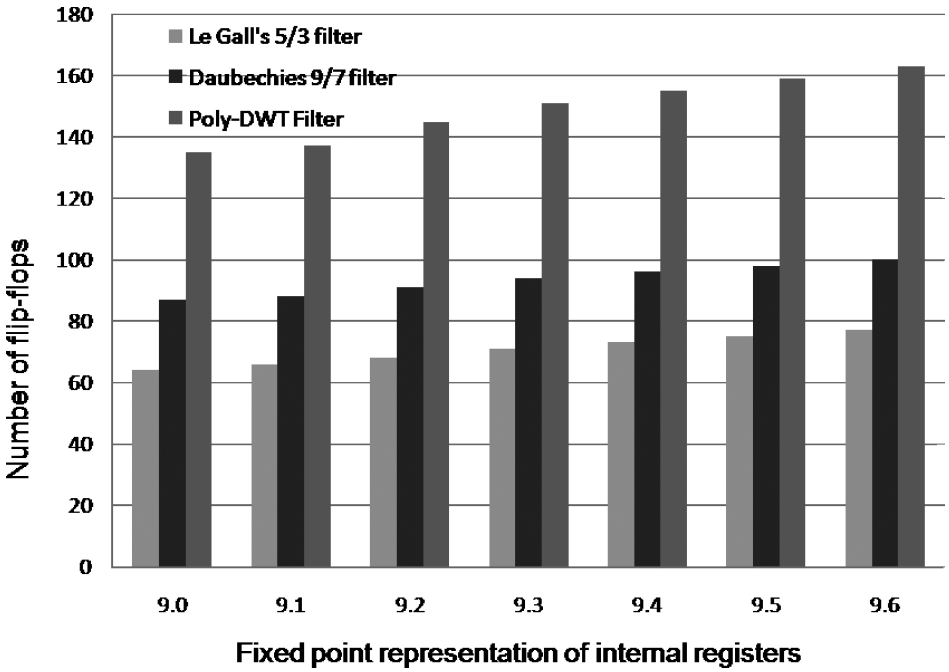


Fig. 12. Comparison of register usage for the binary filter implementations.

23.87 miles. A surveillance system featuring 29 cameras mounted at different points along the bridge is used to keep guard with cameras placed at approximately every 3 miles. Employees monitor the bridge traffic with the help of this system. We propose a dynamic power-saving solution using Poly-DWT considering the usage of surveillance cameras. There are two usages associated with these cameras.

(1) *Idle-usage.* Most of the time, the cameras are used for monitoring the traffic and a low resolution version of these 29 images is provided to the users. Essentially, a very coarse version of the input video is provided to the employees at monitoring station.
(2) *Active-usage.* When a suspected activity is detected, the employee scans for a high-resolution version of the video. A high-resolution version of the surveillance video from concerned video camera is sought. This may be the case of traffic congestion, or someone trying to commit suicide or a car broken down midway in the bridge.

The 9/7 poly-DWT filter has higher hardware requirements and hence consumes more power than the 5/3 filter. Using Xilinx Xpower analyzer for our Xilinx Virtex-5 FPGA, we obtain a power consumption of 0.34W for the 5/3 filter and 0.46W for 9/7 filter. Using the poly-DWT filter during active-usage time and switching to the 5/3 filter during idle-usage time will save us 0.12W power. The respective values were 0.0477 W for 5/3 filter and 0.06 W for 9/7 filter using low power Xilinx Spartan FPGAs.

Most of the time (nearly 99 percent of time) is idle-time for each camera. We get a power saving by a factor of $\frac{0.46}{0.46\times0.99+0.34\times0.01} = 1.348$ (for Virtex-5) and by using our poly-DWT filter.

Another practical scenario is the usage of speed cameras for monitoring traffic. Speed cameras use several different types of technology, most commonly lasers or radar, to pinpoint cars that are exceeding the marked speed limit. When a speeding car is detected, the radar or laser signal triggers the camera to record the car's license plate and that data is used to issue a ticket to the car's owner. Reading the number plate requires a DWT filter with large taps such as the 9/7 filter. On the other hand, the normal usage of camera can be to monitor traffic (at coarse resolution) which is served better by 5/3 filter. The 9/7 poly-DWT filter can be used to get a more accurate view of car's license plate when triggered by radar/laser signal triggers while we can switch to Le Gall's 5/3 filter for keeping a record of traffic movements and also make power-savings.

Time-crucial surveillance applications such as meteorology, remote scientific experiments, defense applications require such rapid switching (in one-two cycles as provided by Poly-DWT) of the hardware architectures.

8. CONCLUSIONS AND FUTURE WORK

This article introduces the concept of polymorphic wavelet architecture for image processing and compression. Polymorphism allows for real-time implementations to dynamically configure the device to allocate hardware resources to suit its instantaneous needs and obtain an area/power optimized design. We presented a low hardware (binary rational) implementation of Daubechies 9/7 filter and its derivation from Le Gall's 5/3 filter outputs to allow on the fly switching between the transform structures upon the demands of application. Moreover, a study of filter performance with the changes in word width allocation was performed. We discussed how internal hardware resource allocation for computational purpose changes the area/reconstruction quality performance of the DWT kernel. The experiments favored the theory of polymorphic wavelet architecture design for dynamic image compression applications.

As a future work, such architectures can be developed for other image compression modules. Moreover, most aspects of DWT implementation and dynamic reconfiguration can be explored further. For example—the number of DWT kernels utilized in image transform and the multiplexing between row and column kernels can be studied to add yet another dimension of polymorphism to our architecture.

REFERENCES

ADAMS, M. AND KOSSENTINI, F. 2000. JasPer: a software-based JPEG-2000 codec implementation. In *Proceedings of the IEEE International Conference on Image Processing (ICIP '00) 2*, 53–56.

ALAM, M., RAHMAN, C., BADAWY, W., AND JULLIEN, G. 2003. Efficient distributed arithmetic based dwt architecture for multimedia applications. In *Proceedings of the International Workshop on SoC for Real Time Applications*. 333–336.

ANSARI, R., GUILLEMOT, C., AND KAISER, J. 1991. Wavelet construction using Lagrange halfband filters. *IEEE Trans. Circ. Syst. 38,* 9, 1116–1118.

BENKRID, A., BENKRID, K., AND CROOKES, D. 2003. Design and implementation of a generic 2D orthogonal discrete wavelet transform on FPGA. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 162–172.

BENKRID, A., CROOKES, D., AND BENKRID, K. 2001. Design and implementation of a generic 2D biorthogonal discrete wavelet transform on an FPGA. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 190–198.

CHANG, M. AND HAUCK, S. 2004. Automated least-significant bit datapath optimization for FPGAs. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 59–67.

CHOI, S.-J. AND WOODS, J. 1999. Motion-compensated 3-*d* subband coding of video. *IEEE Trans. Image Proc. 8,* 2, 155–167.

CHRISTOPOULOS, C., SKODRAS, A., AND EBRAHIMI, T. 2000. The JPEG2000 still image coding system: an overview. *IEEE Trans. Consum. Electron. 46,* 4, 1103–1127.

CHRYSAFIS, C. AND ORTEGA, A. 1998. Line based reduced memory, wavelet image compression. In *Proceedings of the Data Compression Conference, 1998, (DCC '98)*. 398–407.

CLAUS, C., STECHELE, W., KOVATSCH, M., ANGERMEIER, J., AND TEICH, J. 2008a. A comparison of embedded reconfigurable video-processing architectures. In *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications (FPL '08)*, 587–590.

CLAUS, C., ZHANG, B., STECHELE, W., BRAUN, L., HUBNER, M., AND BECKER, J. 2008b. A multi-platform controller allowing for maximum Dynamic Partial Reconfiguration throughput. In *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications (FPL '08)*, 535–538.

EECKHAUT, H., DEVOS, H., LAMBERT, P., SCHRIJVER, D. D., LANCKER, W. V., NOLLET, V., AVASARE, P., CLERCKX, T., VERDICCHIO, F., CHRISTIAENS, M., SCHELKENS, P., DE WALLE, R. V., AND STROOBANDT, D. 2007. Scalable, wavelet-based video: From server to hardware-accelerated client. *IEEE Trans. Multimed. 9,* 7, 1508–1519.

FRY, T. AND HAUCK, S. 2005. SPIHT image compression on FPGAs. *IEEE Trans. Circuits Syst. Video Technol. 15,* 9, 1138–1147.

GALL, D. L. AND TABATABAI, A. 1988. Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 761–764.

HUANG, C.-T., TSENG, P.-C., AND CHEN, L.-G. 2003. VLSI architecture for discrete wavelet transform based on B-spline factorization. In *Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS)*. 346–350.

KOCH, D., BECKHOFF, C., AND TEICH, J. 2008. ReCoBus-Builder - A novel tool and technique to build statically and dynamically Reconfigurable Systems for FPGAs. In *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications (FPL '08)*.

KOTTERI, K., BARUA, S., BELL, A., AND CARLETTA, J. 2005. A comparison of hardware implementations of the biorthogonal 9/7 DWT: convolution versus lifting. *IEEE Trans. Circ. Syst. II 52,* 5, 256–260.

LEESER, M., MILLER, S., AND HAIQIAN, Y. 2004. Smart camera based on reconfigurable hardware enables diverse real-time applications. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 147–155.

LIAN, C.-J., CHEN, K.-F., CHEN, H.-H., AND CHEN, L.-G. 2003. Analysis and architecture design of block-coding engine for EBCOT in JPEG 2000. *IEEE Trans. Circ. Syst. Video Technol. 13,* 3, 219–230.

MARTINA, M. AND MASERA, G. 2005. Low-complexity, efficient 9/7 wavelet filters implementation. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*.

MARTINA, M. AND MASERA, G. 2007. Multiplierless, folded 9/7 - 5/3 wavelet VLSI architecture. *IEEE Trans. Circuits Syst. II 54,* 9, 770–774.

MITTAL, A., PANDE, A., AND VERMA, P. K. 2007. Content-based network resource allocation for mobile engineering laboratory applications. In *Proceedings of the International Conference on Mobile Learning*. 146–152.

PANDE, A., VERMA, A., MITTAL, A., AND AGRAWAL, A. 2007. Network aware efficient resource allocation for mobile-learning video systems. In *Proceedings of the International Conference on Mobile Learning*. 189–196.

PANDE, A. AND ZAMBRENO, J. 2008a. Design and analysis of efficient reconfigurable wavelet filters. In *Proceedings of the IEEE International Conference on Electro Information Technology*. 337–342.

PANDE, A. AND ZAMBRENO, J. 2008b. Polymorphic wavelet architecuture over reconfigurable hardware. In *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications*. 471–474.

PAULSSON, K., HUBNER, M., AND BECKER, J. 2008. Exploitation of dynamic and partial hardware reconfiguration for on-line power/performance optimization. In *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications (FPL '08)*. 699–700.

QIU, R. AND YU, W. 2001. An efficient quality scalable motion-JPEG2000 transmission scheme. Tech. rep. WUCS-01-37, Department of Computer Science, Washington University in St. Louis. Nov.

REDMILL, D., BULL, D., AND MARTIN, R. 1997. Design of multiplier free linear phase perfect reconstruction filter banks using transformations and genetic algorithms. In *Proceedings of the International Conference Image Processing and Its Applications*.

RITTER, J. AND MOLITOR, P. 2001. A pipelined architecture for partitioned DWT based lossy image compression using FPGAs. In *Proceedings of the International Symposium on Field Programmable Gate Arrays (FPGA)*. 201–206.

SAID, A. AND PEARLMAN, W. 1996. An image multiresolution representation for lossless and lossy image compression. *IEEE Trans. Image Process. 5*, 1303–1310.

SCHWARZ, H., MARPE, D., AND WIEGAND, T. 2007. Overview of Process. video coding extension of the H.264/AVC standard. *IEEE Trans. Circ. Syst. Video Technol. 17,* 9, 1103–1120.

SHAPIRO, J. 1993. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. Signal Process. 41,* 12, 3445–3462.

SKODRAS, A., CHRISTOPOULOS, C., AND EBRAHIMI, T. 2001. The JPEG 2000 still image compression standard. *IEEE Signal Proc. Mag. 18,* 5, 36–58.

STINE, J., CASTELLANOS, I., WOOD, M., HENSON, J., LOVE, F., DAVIS, W., FRANZON, P., BUCHER, M., BASAVARAJAIAH, S., OH, J., AND JENKAL, R. 2007. FreePDK: An open-source variation-aware design kit. In *Proceedings of the IEEE International Conference on Microelectronic Systems Education*, i–iii.

STROOBANDT, D., EECKHAUT, H., DEVOS, H., CHRISTIAENS, M., VERDICCHIO, F., AND SCHELKENS, P. 2004. Reconfigurable hardware for a scalable wavelet video decoder and its performance requirements. *Computer Systems: Architectures, Modeling, and Simulation 3133*, 203–212.

TAUBMAN, D. 2000. High performance scalable image compression with EBCOT. *IEEE Trans. Image Process. 9,* 7, 1158–1170.

TAY, D. 2000. Rationalizing the coefficients of popular biorthogonal wavelet filters. *IEEE Trans. Circuits Syst. Video Technol. 10,* 6, 998–1005.

TIAN, J. 2008. SPIHT coder. http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4808&objectType=file.

TSENG, P., CHANG, Y., HUANG, Y., FANG, H., HUANG, C., AND CHEN, L. 2005. Advances in hardware architectures for image and video coding - A survey. *Proc. IEEE 93,* 1, 184–197.

VERMA, P. K., MITTAL, A., AND KUMAR, P. 2006. Fusion of thermal infrared and visible spectrum video for robust surveillance. In *Proceedings of ICVGIP*. 528–539.

VERMA, P. K., PANDE, A., MITTAL, A., AND KUMAR, P. 2008. Content-based network adaptive wireless transmission of remote surveillance video. In *Proceedings of the National Conference on Communications*.

VETTERLI, M. AND KOVAČEVIC, J. 1995. *Wavelets and Subband Coding*. Prentice-Hall, Inc., Upper Saddle River, NJ.

VILLASENOR, J., BELZER, B., AND LIAO, J. 1995. Wavelet filter evaluation for image compression. *IEEE Trans. Image Process. 4,* 8, 1053–1060.

YANG, W., LU, Y., WU, F., CAI, J., NGAN, K. N., AND LI, S. Nov. 2006. 4-D Wavelet-Based Multiview Video Coding. *IEEE Trans. Circuits Syst. Video Technol. 16,* 11, 1385–1396.

ZHANG, X., RABAH, H., AND WEBER, S. 2007. Auto-adaptive reconfigurable architecture for scalable multimedia applications. In *Proceedings of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 139–145.

Received December 2008; revised May 2009 and December 2009; accepted December 2009