# POLYMORPHIC WAVELET ARCHITECTURES USING RECONFIGURABLE HARDWARE

*Amit Pande and Joseph Zambreno*

Department of Electrical and Computer Engineering
Iowa State University, Ames, IA-50011, USA
email: {amit, zambreno}@iastate.edu

## ABSTRACT

Traditional microprocessor-based solutions are insufficient to serve the dynamic throughput demands of real-time scalable multimedia processing systems. This paper introduces a Polymorphic Architecture for the Discrete Wavelet Transform (Poly-DWT) as a building block of reconfigurable systems to address these needs. We illustrate how our Poly-DWT architecture can dynamically make resource allocation decisions according to application requirements. We perform a quantitative analysis of our Poly-DWT architecture using an FPGA prototype, and compare our filters to existing approaches to illustrate the area and performance benefits inherent in our approach."

## 1. INTRODUCTION

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 has been used to provide time-frequency analysis of images which allows scalable encoding of multimedia [1, 2, 3]. The large computational complexity and memory requirements involved in real-time image processing algorithms have been a bottleneck for such systems. Previous hardware implementations such as those using ASICs or FPGAs are capable of accelerating these computations by exploiting the inherent algorithmic parallelism. Hardware implementations [4, 5, 6] aim at reducing hardware complexity in order to improve system throughput.

In this paper, 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.

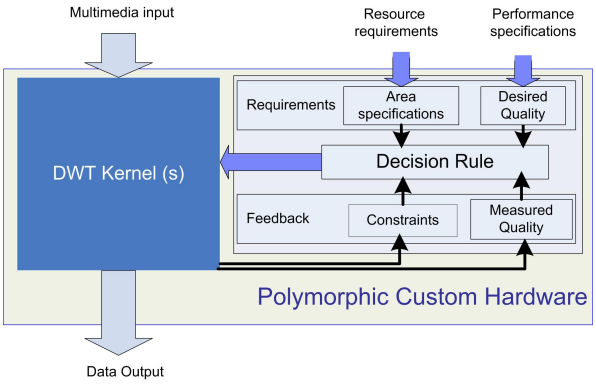


**Fig. 1**. Conceptual Overview of the Polymorphic Wavelet Architecture

Figure 1 gives a general description of Poly-DWT and its interface with a larger multimedia system. Multimedia input such as stream of pixels is first transformed into the time-frequency domain by the wavelet transform. An interface is provided for the application to dynamically notify the architecture about its performance requirements in terms of the hardware constraints and the image reconstruction quality demands. The hardware resources allocated for the computation of DWT can be reconfigured to exploit the inherent tradeoff between the device area and power usage versus the image compression performance of the architecture.

The contributions of this paper are summarized as follows:

- We introduce the concept of the Polymorphic Discrete Wavelet Transform (Poly-DWT) architecture to enable dynamic reconfiguration of hardware resources.
- We present a quantitative analysis of the various factors and tradeoffs involved in a Poly-DWT implementation. A multiplier-free architecture is obtained to enable higher clock speed.
- We present results of FPGA synthesis oof Poly-DWT filters.

**Table 1**. Analysis and Synthesis Filter Coefficients

| | Daubechies 9/7 Filter | | Poly-DWT Filter | |
|---|---|---|---|---|
| $i$ | $h_0(i)$ | $h_1(i)$ | $h_0(i)$ | $h_1(i)$ |
| $\pm 4$ | 0.026748757411 | 0 | $1/64$ | 0 |
| $\pm 3$ | -0.016864118443 | 0.091271763114 | 0 | $1/32$ |
| $\pm 2$ | -0.078223266529 | -0.057543526229 | $-1/8$ | 0 |
| $\pm 1$ | 0.266864118443 | -0.591271763114 | $-1/4$ | $-9/32$ |
| 0 | 0.602949018236 | 1.11508705 | $23/32$ | $1/2$ |

- Given the image quality constraints, the Poly-DWT 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.

## 2. 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 efficient wavelet transform filters for high-quality compression of image or video data and power efficient design.

Biorthogonal Wavelet Filter Banks (BWFBs) are commonly used for image processing. Due to irrational values in their filter coefficients, any associated hardware implementation requires a high precision architecture. Many research works have faced the problem of reducing the DWT complexity [7, 8, 5]. The most common image processing wavelet filter is CDF-9/7, which is accepted as the standard for lossless compression in JPEG2000 [3]. The implementation of CDF-9/7 over fixed point arithmetic leads to quantization error and requires large amounts of hardware resources for perfect image reconstruction. Recently, multiplier-free implementations have been proposed to reduce the computational complexity [5, 6]. Most implementations approximate the irrational coefficients into nearby rational values, leading to image reconstruction quality tradeoffs.

In [9], the authors present a technique to rationalize the coefficients of wavelet filters that will preserve biorthogonality and perfect reconstruction. This approach also preserves regularity of the structure. In this paper, we exploit this technique to obtain the various binary coefficient multiplier-free implementations of the 9/7 filters in this paper.

## 3. MOTIVATION AND INSIGHT

In this section we discuss some dimensions that provide this polymorphic capacity to our architecture. The first dimension is the choice of suitable DWT filter to serve instance and application specific requirements. The complexity of the DWT hardware is another important aspect. An implementation with diverse hardware requirements such as multipliers and buffers is less amenable to dynamic reconfiguration. Diverse hardware requirements also imply larger power, and slower performance.

The two most common and standard filters for DWT over images are Le Gall's 5/3 filter and the Daubechies 9/7 filter [3]. Poly-DWT filter tries to achieve high image compression like Daubechies filter with cheaper implementation cost, higher throughput, and on-the-fly switching to a 5/3 filter architecture.

There are four filters that comprise the two-channel biorthogonal wavelet system. The analysis and synthesis low-pass filters are denoted by $H_0$ and $G_0$, respectively.

A look at first two columns in Table 1 sheds light on 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 the filter coefficients. Moreover, hardware multipliers would be needed to implement this in our design with reasonable precision. Tay [9] increases the number of freedom in original expression for Daubechies bi-orthogonal filter bank derivation and utilizes this to obtain expression for binary rational coefficients for 9/7 filters. A poly-DWT design is built over such rational coefficients filters obtained by a numerical study and analysis to achieve low quantization loss [10]. The filter coefficients are given in Table 1.We use this filter in this paper to illustrate some advantages of our Poly-DWT architecture.

### 3.1. On the Fly Switching between 9/7 and 5/3 Filter Structures

One motivation behind our Poly-DWT architecture is to provide dynamic switching between wavelet filters. Le Gall's 5/3 filter has rational coefficients and has been used in image processing as a standard for image compression [3]. In this section we develop a structure to allow dynamic switching between the poly-DWT 9/7 filter and Le Gall's 5/3 filter.

In [10] we present the mathematics to reach at the coefficients for the Poly-DWT filter. We use those filters to reach the following relation:

$$low97(i) = low53(i) - \frac{w(0)}{32} + \frac{w(4)}{64}$$

$$high97(i) = \frac{high53(i)}{2} - \frac{w(1)}{32} + \frac{w(3)}{32}$$

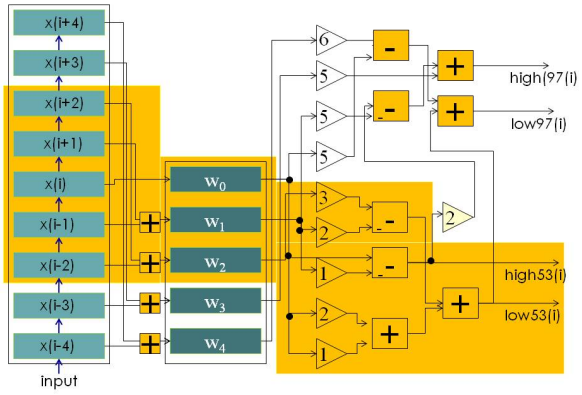


**Fig. 2**. An example of Poly-DWT filter architecture

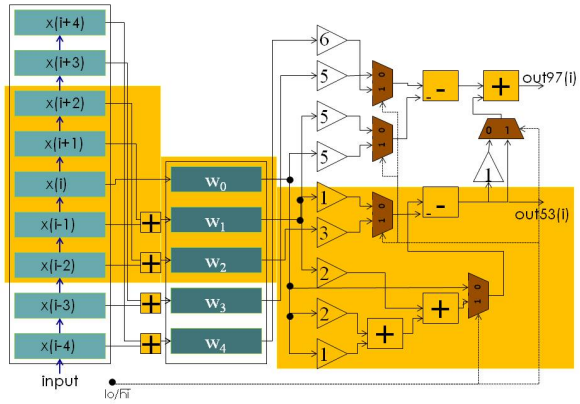


**Fig. 3**. An example of folded implementation of Poly-DWT filter architecture

where $low53(i)$ and $high53(i)$ are the Le Gall's filters and $low97(i)$ and $high97(i)$ are Poly-DWT filters.

It is evident that we can develop a perfect reconstruction Poly-DWT filter with binary coefficients. This implementation is illustrated in the block diagram given in Fig. 2. The yellow/ shaded region is the architecture common to the 5/3 and 9/7 filters.

The division by $2, 4, 8, 32$, and $64$ can be implemented using shift operations. As illustrated in Fig. 2, the Le Gall's 5/3 filter implementation requires only eight adder/subtractor units. The binary 9/7 Poly-DWT filter requires only twelve adder/ subtractor units.

### 3.2. Power and Space Efficient Folded Design

DWT output signals can be down-sampled by a factor of two without any loss of information. This motivates us to further reduce the hardware cost of implementation. We can multiplex the hardware resources to obtain a more efficient design which is referred to as folded architecture. As shown in Fig. 3, our folded architecture implements Le Gall's 5/3 filter in five adder/subtractor units while our Poly-DWT filter is implemented using nine adder/subtractor units. Signal $out97(i)$ has signals $low97(i)$ and $high97(i)$ multiplexed into it using a 2x1 multiplexer in the hardware. By changing the values of the signal $lo/hi$ alternatively every other cycle we can get the high and low pass values. The dark shaded (yellow) regions are the common hardware resources for the 5/3 and 9/7 filter implementation. When the Poly-DWT implements a 5/3 filter, the remaining hardware is switched off to save power.

Other possible aspects of the Poly-DWT implementation (number of DWT kernels required, bit width of internal representation of registers, separation / folding of row and column processing DWT kernels etc) have not been discussed for brevity and they are left as a future work.

### 3.3. Fixed Point Description

In this work, we avoid the floating-point implementation of the system to prevent a non-optimal usage of resources. There are two conflicting issues that affect the decision to decide the hardware bit allocation for internal representation of variables:

(a) Increased number of bits generally implies better performance in terms of image quality and reduced error.

(b) Decreased number of bits imply a better hardware utilization, and lower power consumption.

A dynamic tradeoff in power versus reconstruction quality is required to cater to different applications. The experiments for simulating variable bit width of internal registers used show an interesting trend as explained in experimentation section.

## 4. EXPERIMENTAL ANALYSIS

We targeted a Xilinx XC5VLX30 FPGA for our experiments, using ModelSim 6.0c for simulation purposes and Xilinx ISE 9.1i for synthesis.

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 in fixed point hardware implementation. We represented the intermediate variables in Q9.4 fixed point format for this experimental setup. The more detailed analysis of image reconstruction performance of various filters was done using MATLAB.

Direct implementation of the CDF-9/7 filter gave a clock frequency of 107 MHz, while requiring 16 multiplier units. Martina et al. [6] report a clock frequency of about 200 MHz through a multiplier-free implementation, targeting 0.13 $\mu m$

**Table 2**. Comparison of binary filter features and hardware resources requirements

| Features | Daubechies 9/7 Filter | Poly-DWT 9/7 Filter | Folded Poly-DWT | [11] | [4] | [12] | [6] | [5] |
|---|---|---|---|---|---|---|---|---|
| Adders | 10 | 12 | 9 | 19 | 15 | 8 | 19 | 21 |
| Multipliers | 16 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| PSNR grade | A | A | A | C | C | A | B | B |
| Reconfigurable? | N | Y | Y | N | N | N | Y | N |

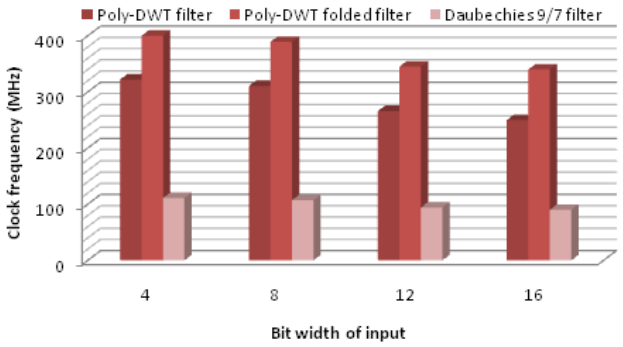


**Fig. 4**. Change in FPGA clock frequency (MHz) for variable bit widths for various filters

VLSI technology. By pipelining the individual shift and add operations, we were able to achieve vey high clock frequencies (over 350 MHz on our target Virtex-5 platform) for our Poly-DWT design. The change in frequency versus bit width of internal representation and type of filter is shown in figure 4.

The hardware resources utilized in these DWT kernels are summarized in Table 2. A comparison of hardware resources utilization is compared to existing works.

FPGA-specific implementation results are given in Table 3. Here, the effect of change in bit width of input data on the hardware resource usage of the FPGA chip can be seen. Poly-DWT requires no multipliers, leading to an overall efficient area/power profile than original filters.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper we discussed the hardware design of a Polymorphic Wavelet Architecture to enable dynamic real-time multimedia applications. A multiplier-free implementation of DWT requiring only nine adders was synthesized on a Xilinx Virtex V FPGA and achieved a clock frequency of over 350 MHz while allowing on-the-fly switching to a simpler DWT filter.

The future work includes adding new dimensions of parallelism to the architecture including bit width variations and memory access.

**Table 3**. FPGA resource utilization comparison (BM = Block Multipliers, LS = Logic Slices)

| Bidwidth | Le Gall 5/3 Filter | | Poly-DWT Filter | | Daubechies 9/7 Filter | |
|---|---|---|---|---|---|---|
| | BM | LS | BM | LS | BM | LS |
| 4 | 0 | 24 | 0 | 103 | 9 | 35 |
| 8 | 0 | 75 | 0 | 181 | 9 | 96 |
| 12 | 0 | 108 | 0 | 259 | 9 | 132 |
| 16 | 0 | 152 | 0 | 337 | 9 | 176 |

## 6. REFERENCES


[1] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[2] A. Said and W. Pearlman, "An image multiresolution representation for lossless and lossy image compression," *IEEE Transactions on Image Processing*, vol. 5, pp. 1303–1310, 1996.

[3] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 1103–1127, Nov 2000.

[4] K. Kotteri, S. Barua, A. Bell, and J. Carletta, "A comparison of hardware implementations of the biorthogonal 9/7 DWT: convolution versus lifting," *IEEE Transactions on Circuits and Systems II*, vol. 52, no. 5, pp. 256–260, May 2005.

[5] M. Martina and G. Masera, "Low-complexity, efficient 9/7 wavelet filters implementation," in *Proceedings of the Intl. Conference on Image Processing (ICIP)*, Sept. 2005.

[6] ——, "Multiplierless, folded 9/7 5/3 wavelet VLSI architecture," *IEEE Transactions on Circuits and Systems II*, vol. 54, no. 9, pp. 770–774, Sept. 2007.

[7] M. Alam, C. Rahman, W. Badawy, and G. Jullien, "Efficient distributed arithmetic based dwt architecture for multimedia applications," in *Proceedings of the Intl. Workshop on SoC for Real Time Applications*, 2003, pp. 333–336.

[8] J. Ritter and P. Molitor, "A pipelined architecture for partitioned dwt based lossy image compression using FPGAs," in *Proceedings of the Intl. symposium on Field Programmable Gate Arrays (FPGA)*, 2001, pp. 201–206.

[9] D. Tay, "Rationalizing the coefficients of popular biorthogonal wavelet filters," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 10, no. 6, pp. 998–1005, Sept. 2000.

[10] A. Pande and J. Zambreno, "Design and analysis of efficient reconfigurable wavelet filters," in *IEEE International Conference on Electro/ Information Technology*.

[11] D. Tay, "A class of lifting based integer wavelet transform," in *Proceedings of the Intl. Conference on Image Processing (ICIP)*, 2001, pp. 602–605.

[12] C. Huang, P. Tseng, and L. Chen, "Flipping structure: an efficient VLSI architecture for lifting-based discrete wavelet transform," *IEEE Transactions on Signal Processing*, vol. 52, no. 4, pp. 1080–1089, Apr. 2004.