A Dual Mode (5, 3)/(9,7) FDWT/IDWT Hardware Accelerator IP

Chih-Chun Chang

Dept CS
National Tsing Hua Univ
Hsin-Chu, Taiwan 300
Tel : +886-3-5742797
Fax : +886-3-5731072
dajovu@nthucad.cs.nthu.edu.tw

Youn-Long Lin

Dept CS
National Tsing Hua Univ
Hsin-Chu, Taiwan 300
Tel : +886-3-5731072
Fax : +886-3-5731072
ylin@cs.nthu.edu.tw

Abstract - We propose a software tool for automatic generation of hardware accelerators for performing Discrete Wavelet Transform (DWT) with various coefficient parameters. In addition to (5, 3) and (9, 7) DWT filters defined in the JPEG2000 image compression standard, other filters such as (9, 3), (6, 10) and (2, 2) can also be generated. Our generated hardware IP can perform both forward and inverse transform (FDWT and IDWT). Our tool also produces on-chip-bus interface circuit compliant with the AMBA protocol and associated device driver so that the generated IPs are ready for SOC integration. We verify the proposed approach by integrating generated IPs into an SOC platform running JPEG2000 application software. Experimental results demonstrated that the proposed approach is indeed effective in enhancing the productivity of hardware accelerator IP design. The generalization portion is omitted from this paper due to page limitation.

I. Introduction

For multimedia processing, Discrete Wavelet Transform (DWT)-based image coding has better performance than traditional DCT-based image coding, especially for low bit-rate applications. Therefore, DWT-based approach has been adopted for next generation image coding standards such as JPEG2000 [8], Motion-JPEG2000 [13], and MPEG4 still image coding [14]. Because the DWT function accounts for a significant portion of the encoding/decoding time, it is advantageous to speed up the function with dedicated hardware accelerator. For easy SOC integration, a hardware accelerator should be compliant with popular on-chip-bus protocols.

In practice, different standards use DWT filters with different coefficient. For example, we use (5, 3) and (9, 7) filters in JPEG2000 and (9, 3) filter in MPEG4. A parameterized DWT IP generator will greatly increase our productivity.

A typical DWT-based image coding uses DWT to transform image data into wavelet coefficients first. After the quantized wavelet coefficients are processed through the entropy coder, the final compressed image is produced. DWT processes in larger scale to eliminate blocking artifacts suffered by DCT-based image coding. DWT can be done in either lossy or lossless mode. If we choose appropriate FDWT / IDWT pairs, perfect reconstruction is possible. The most important propriety of DWT is multi-resolution decomposition that can achieve low bit rate and high quality.

The rest of this paper is organized as following. Section II describes the wavelet transform theory and surveys some architectures proposed previously. Section III describes basic architectures for both forward DWT and inverse DWT with emphasis on reducing the power consumption of critical modules. Section IV gives the important lifting-based DWT theory and its corresponding architecture. In Section V, we present our synthesis, DFT, and ATPG result. In Section VI, we describe our IP integration and verification environment. We then present an AMBA-based DMA interface for the IP in Section VII. Finally, we draw some concluding remarks and point to possible directions for future research in Section VIII.

II. Previous Work

Much work has been performed on DWT theory and implementation. Mallat combined the Wavelet transform and filter bank into a single transformation [15]. Doubechie applies DWT to image coding and proposed many famous wavelet filters [16] including the (9, 7) filter. Swendens proposed the Lifting Scheme (LS) [5] making DWT more computationally efficient. Calderbank, Doubechies and Swendens later proposed the Integer Wavelet Transform (IWT) [17] that is more efficient without scarifying the performance. Vishwanath proposed the Recursive Pyramid Algorithm (RPA) [2] and a systolic array implementation. Ferretti proposed a modified RPA [] to solve the boundary problem encountered during perfect reconstruction.

The lifting scheme is adopted by the JPEG2000 standard finalized in March 2000. Many lifting-based DWT architectures have been proposed. Grangetto, Magli, and Martina proposed a criterion for optimal factorization of DWT [18]. Fig. 1 depicts the DWT functionality of JPEG2000.

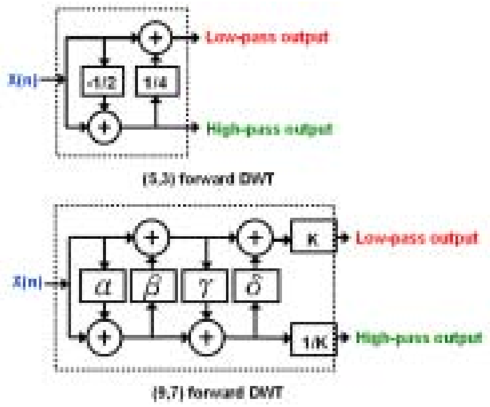


Fig. 1 The DWT functionality in JPEG2000

Chen et al [4] proposed a combined (5, 3) filter and (9, 7) filter architecture by means of folding as shown in Fig. 2.

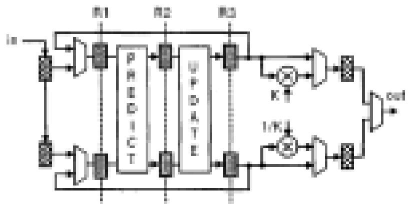


Fig. 2 Chen's Combined (5, 3) and (9, 7) Filter Architecture

III. DWT Architecture

In this section, we improve Chen's architecture so that it can perform both FDWT and IDWT. After estimating the power consumption of our DWT architecture, we identify the hot spot and reduce the power consumption. Furthermore, we improve the circuit performance by means of register retiming.

A. Combined forward DWT and inverse DWT

We propose a dual mode architecture that can perform both FDWT and IDWT. First observing the (5, 3) filter FDWT and IDWT described in the lifting scheme, we find that many components are common to both directions. The (9, 7) filter also has similar characteristics.

$$Y(2n+1) = X_{ext}(2n+1) - \left\lfloor \frac{X_{ext}(2n) + X_{ext}(2n+2)}{2} \right\rfloor$$
$$Y(2n) = X_{ext}(2n) + \left\lfloor \frac{Y(2n-1) + Y(2n+1) + 2}{4} \right\rfloor$$

(5, 3) filter Forward Discrete Wavelet Transform

$$X(2n) = Y_{ext}(2n) - \left\lfloor \frac{Y_{ext}(2n-1) + Y_{ext}(2n+1) + 2}{4} \right\rfloor$$
$$X(2n+1) = Y_{ext}(2n+1) + \left\lfloor \frac{X(2n) + X(2n+2)}{2} \right\rfloor$$

(5, 3) filter Inverse Discrete Wavelet Transform

Fig. 3 The (5, 3) wavelet operation in the lifting scheme

For the (5, 3) filter, the division by four or by two can be easily implemented with shifting two bits or one bit in the predict module or the update module. But for the (9, 7) filter, the operation is more complex. We still can find some common operation and reuse the predict module, the update module and the K module (scaling by K or 1/K).

$$\begin{cases} Y(2n+1) \leftarrow X_{ext}(2n+1) + (\alpha \times [X_{ext}(2n) + X_{ext}(2n+2)]) & [STEP1] \\ Y(2n) \leftarrow X_{ext}(2n) + (\beta \times [Y(2n-1) + Y(2n+1)]) & [STEP2] \\ Y(2n+1) \leftarrow Y(2n+1) + (\gamma \times [Y(2n) + Y(2n+2)]) & [STEP3] \\ Y(2n) \leftarrow Y(2n) + (\delta \times [Y(2n-1) + Y(2n+1)]) & [STEP4] \\ Y(2n+1) \leftarrow -K \times Y(2n+1) & [STEP5] \\ Y(2n) \leftarrow (1/K) \times Y(2n) & [STEP6] \end{cases}$$

(9, 7) filter Forward Discrete Wavelet Transform

$$\begin{cases} X(2n) \leftarrow K \times Y_{ext}(2n) & [STEP1] \\ X(2n+1) \leftarrow -(1/K) \times Y_{ext}(2n+1) & [STEP2] \\ X(2n) \leftarrow X(2n) - (\delta \times [X(2n-1) + X(2n+1)]) & [STEP3] \\ X(2n+1) \leftarrow X(2n+1) - (\gamma \times [X(2n) + X(2n+2)]) & [STEP4] \\ X(2n) \leftarrow X(2n) - (\beta \times [X(2n-1) + X(2n+1)]) & [STEP5] \\ X(2n+1) \leftarrow X(2n+1) - (\alpha \times [X(2n) + X(2n+2)]) & [STEP6] \end{cases}$$

(9, 7) filter Inverse Discrete Wavelet Transform

Fig. 4 The (9, 7) wavelet operation in the lifting scheme

By adding some routing paths and multiplexers as depicted in the Figure below, we can perform both FDWT and IDWT using the same architecture. Compared the architecture proposed by Chen [4], our architecture needs only six additional multiplexers to gain the dual mode capability.

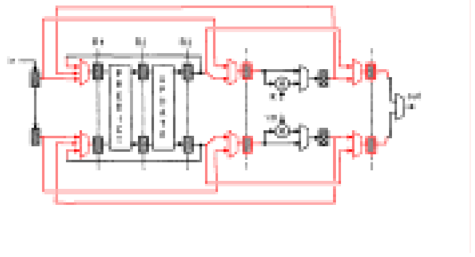


Fig. 5 The proposed Dual Mode FDWT/IDWT Architecture

B. The bit precision

Let's focus on how many bits as needed to preserve the precision. We will discuss the integer bits and the fraction bits separately. The input to the DWT module is the YCrCb component. Each pixel of a YCrCb component is about 8 bits or 9 bits. By tracing the expression of the (9, 7) filter forward DWT, we find that the output could be 8.265234 times larger than the input in the worst case. Therefore, we need 5 more bits including the guard bits. Therefore, we use 14 bits for the integer bits.

In reference [17], the PSNR (db) value of image Lena for different rate (bpp) and different fraction bits (m) is studied. We learned from it that using 8 bits for the fraction bits has great quality. In order to get higher quality we use 10 bits for fraction bits that will approximate to the infinite fraction bits.

TABLE 1
The bit precision of (9, 7) filter (PSNR)

Daubechies (9, 7) filter			
Fraction bits	0.25 bpp	0.5 bpp	1 bpp
m = ∞	33.6429	36.7810	39.9570
m = 4	23.5472	25.5439	28.3017
m = 8	32.8755	35.5629	38.2426

C. *Low-power and high-performance design*

We already have the FDWT/IDWT architecture. Now we make it more low-power and more high-performance.

C.1 *Improved symmetric extension module*

For symmetric extension of signals at the beginning and ending of input sequence, a shift register module is commonly used. Our power analysis shows that the shifter module consumes about 35% of the total power. By carefully analyzing the life time of each register, we re-design it with a special datapath resulting in 70% power saving in the module.

With the behavior of the symmetric extension, we can use the register minimization techniques from high level synthesis to analyze the lifetime of each input pixel. It is easy to know the minimum number of registers needed. Then we allocate the register to the input. The principle is not to shift the input pixels from registers to registers. Instead we hold input pixels in the same registers and access them through multiplexers.

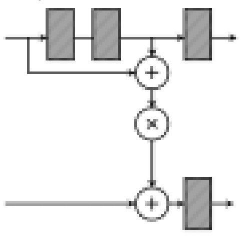
By the allocation table, we can easily use a finite state machine (FSM) to control the data path of nine registers. In our application, we need four FSMs to perform both FDWT and IDWT of both (5, 3) and (9, 7) filters.

C.2 *Canonic Signed Digit (CSD) Multipliers*

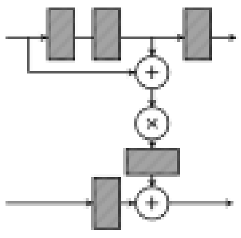
Multiplication is also the critical part of our design. We try to make it more low-power and high-performance. In our application, every coefficients of the multiplication (α, β, γ, δ, K and 1/K) are known already. Therefore we do not need general multipliers. Hartley proposed the canonic signed digit multipliers (CSD) [19] that used shift operation and addition to perform multiplication. For example, to multiply 0.75 is equal to shift right 1 bit plus shift right 2 bits. We can decompose any floating point coefficients into CSD representation. There are some researches in optimizing the CSD representation. Take the K value defined in JPEG2000 (9, 7) filter as example. By the shift line reduction, we can reduce the CSD from nine shift lines to five shift lines. That will reduce both the area and the critical path delay. If a multiplier needs to perform two multiplications, the sub-expression sharing technique is beneficial. We should find the common sub-expression between two or three multiplicands.

C.2 *Using Retiming to improve performance*

After synthesis, we focus on the critical path. We find that the critical path is in the predict module and the update module. A path that performs two additions and one multiplication is critical. We improve it by retiming. We move the registers as depicted in Fig. 6 to shorten the critical path while preserving the functionality. The figure shows the original and the retimed predict module. The critical path delay is reduced from 3.89 ns to 3.10 ns. The update module is also optimized similarily.



The original predict module architecture



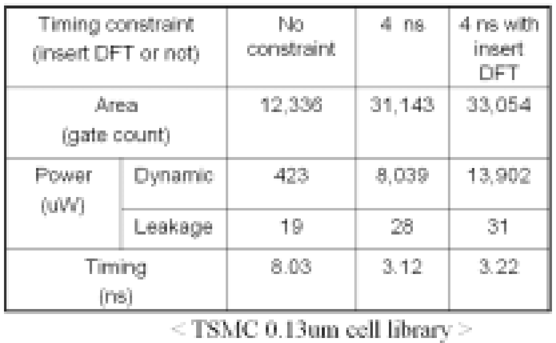
The new predict module architecture after retiming

Fig. 6 Optimizing the predict module by retiming

V. DWT IP Implementation

We have implemented the proposed architecture as a reusable IP. We follow the guidelines defined in the Reuse Methodology Manual (RMM) []. We will give the synthesis report, design-for-test report, and ATPG result.

Our target is to process a 16M-pixel image in 0.5sec. We set the tile size to be 256 * 256 pixels and resolution level 5. We need 152,371,200 clock cycles and clock period of 3.281 ns. Using the TSMC 0.13μm cell library, we are able to achieve this target.



Timing constraint (insert DFT or not)		No constraint	4 ns	4 ns with insert DFT
Area (gate count)		12,336	31,143	33,054
Power (uW)	Dynamic	423	8,039	13,902
	Leakage	19	28	31
Timing (ns)		8.03	3.12	3.22

< TSMC 0.13um cell library >

Fig. 7 Synthesis and insert DFT report

After synthesis, we perform DFT synthesis to insert scan chains. The report is given in Fig. 7. The fault coverage is 98.25% based on the report from TeraMax. We use the tool nLint® to make sure our coding style is compliant with the RMM.

VI. Verification and Integration

To verify our DWT IP, we design a testbench, a bus interface and a device driver, and integrate them all into an

SOC platform for FPGA prototyping.

Our golden model is the official JPEG2000 reference software called Jasper. We extract from Jasper the input and the output of the DWT function. And then we feed the input to our DWT IP and compare the outputs of both software and hardware. We also provide a high code coverage testbench to verify our DWT IP.

We choose the AMBA (Advanced Microcontroller Bus Architecture) AHB (Advanced High-performance Bus) protocol. We use a data-driven architecture. The DWT IP is active only when data arrives.

Our device driver simplifies the use of our DWT IP to a function call as depicted below:

```
DWT(FORWARD,LOSSLESS,DEINTER,256,16,DWT_data,output);
```

Finally, we integrate all software and hardware together in an SOC platform as depicted in. The Jasper software is running on the ARM922T CPU with the DWT function replaced by the function call to the DWT device driver.

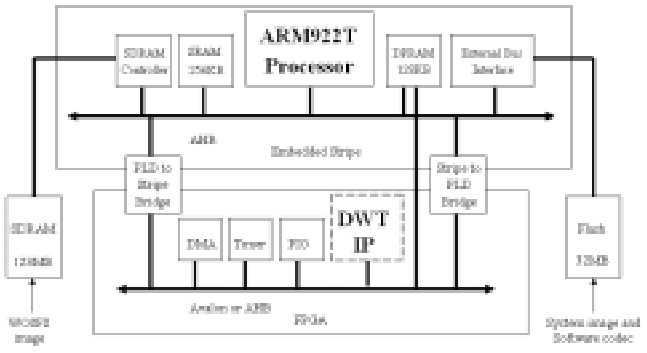


Fig. 8 Integration Environment for JPEG2000 Codec

VII. Optimal AHB interface with DMA

Now we talk about 2D-DWT optimal AMBA (AHB) interface. In the subsection, we also concern about local memory constraints. We consider about two cases. One is that local memory large enough to store the whole tile, the other one is not. And in this case, the flexibility of hardware-software co-design is the point.

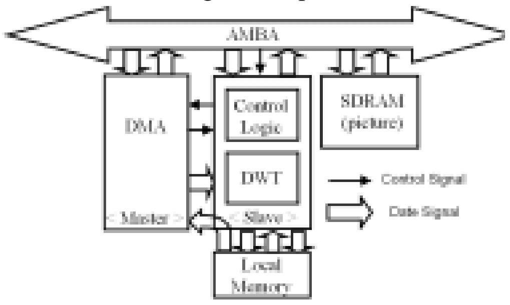


Fig. 9 2D-DWT interface with DMA and Local memory

Fig. 9 is the 2D-DWT interface with large local memory. We use the local memory to store the whole tile. The image of the tile does row wavelet operation, and the wavelet coefficients are stored in the local memory. Then DWT do column wavelet operation to complete 2D-DWT processing. This interface can also gain the benefit to do deinterleave. But the control logic of DMA and the local memory controller are more complicated. The total computation time is (row length)* (2 * row length + 1). The utilization of the bus and our DWT IP are almost 100 %.

VIII. Summary and Conclusions

In this paper, we focus on the generalized DWT architecture. We propose a generalized design for any wavelet filter and factor the coefficient into lifting-scheme automatically. The propose architecture can be fitted for any factoring algorithm. We also focus on the power consumption and try to make the submodule that cost most power more low power.

For the IP concern, we must take care of the IP qualification, the IP verification and the bus interface. The bus interface is something we most care. Using DMA, interrupt, and local memory, we design the optimal interface with hardware-software co-design.

In the future, we can try to integrate the DWT architecture that we generated into other image or video compression system. However, we have already verified the (5, 3) and (9, 7) filter with JPEG2000 image coding system.

References

[1] S. Mallat, “A theory for multi resolution signal decomposition: the wavelet representation,“ IEEE Transactions on Pattern Anal. Machine Intell., Vol.11, pp. 674-693 ,July 1989.
[2] M. Vishwanath, “The Recursive Pyramid Algorithm for the Discrete Wavelet Transform,” IEEE Trans. on Signal Processing, vol. 42, no. 3, pp.673-676, Mar 1994.
[3] C.-Y. Chen, Z.-L. Yang, T.-C. Wang and L.-G. Chen, ” A programmable VLSI architecture for 2-D discrete wavelet transform,” IEEE International Symposium on Circuits and Systems 2000, Vol.1, pp. 619 – 622, May 2000.
[4] C.-T. Huang, P.-C. Tseng and L.-G. Chen, “Efficient VLSI architectures of lifting-based discrete wavelet transform by systematic design method,” IEEE International Symposium on Circuits and Systems 2002, Vol.5, pp. 565 – 568, May 2002.
[5] I. Daubechies and W. Swelden, “Factoring wavelet transform into lifting steps,” The Journal of Fourier Analysis and Applications, Vol. 4, pp.247-269, 1998.
[6] C. Chrysafis and A. Ortega, “Line-based, reduced memory, wavelet image compression,” IEEE Transactions on Image Processing, Vol.9, No.3, pp. 378-389, Mar 2000.
[7] P.-C. Wu and L.-G. Chen, “An efficient architecture for two-dimensional discrete wavelet transform,” IEEE Transactions on Circuits and Systems for Video Technology, Vol.11, Issue.4 , pp. 536-545, April 2001.
[8] ISO/IEC. ISO/IEC 15444-1. Information technology – JPEG2000 Part I. image coding system. Mar 2000.
[9] C.-J. Lian, K.-F. Chen, H.-H. Chen and L.-G. Chen, “Lifting based discrete wavelet transform architecture for JPEG2000,” IEEE International Symposium on Circuits and Systems 2001, Vol.2, pp. 445 – 448, May 2001.
[10] P.-C. Tseng, C.-T. Huang and L.-G. Chen, “Reconfigurable discrete wavelet transform architecture for advanced multimedia systems,” IEEE Workshop on Signal Processing Systems 2003, pp. 137 – 141, Aug 2003.
[11] M. Ravasi, L. Tenze and M. Mattavelli, “A scalable and programmable architecture for 2-D DWT decoding,” IEEE

Transactions on Circuits and Systems for Video Technology, Vol.12, Issue.8 , pp. 671-677, Aug 2002.

[12] K. Andra, C. Chakrabarti, and T. Acharya, “A VLSI architecture for lifting-based forward and inverse wavelet transform,” IEEE Transactions on Signal Processing, Vol. 50, Issue.4 , pp. 966-977, April 2002.

[13] ISO/IEC. ISO/IEC 15444-3. Information technology –Motion JPEG2000. 2002.

[14] ISO/IEC. JTC1/SC29/WG11 Coding of moving pictures and audio. Jan 2001.

[15] A.B. Smith, C.D. Jones, and E.F. Roberts, “Article Title”, Journal, Publisher, Location, pp. 1-10, Date.

[16] A. Cohen, I. Daubechies,and J. Feauveau, “Bi-othogonal bases of compactly supported wavelets“, Comm. Pure Appl. Math., vol. 45, pp. 485-560, 1992.

[17] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, “Wavelet transforms that map integers to integers“, Technical report, Deportment of Mathematics, Princeton University, 1996.

[18] M. Grangetto, E. Magli, M. Martina, and G. Olmo, “Optimization and Implementation of the Integer Wavelet Transform for Image Coding”, IEEE Transactions on Signal Processing, Vol. 11, June 2002.