A Parameterized On-Chip-Bus-Compliant FDWT/IDWT Accelerator IP Generator*

Chih-Chun Chang

Department of Computer Science
National Tsing Hua University
Hsin-Chu, Taiwan 300
Tel : +886-3-574-2797
Fax : +886-3-572-3694
dajovu@nthucad.cs.nthu.edu.tw

Youn-Long Lin

Department of Computer Science
National Tsing Hua University
Hsin-Chu, Taiwan 300
Tel : +886-3-573-1072
Fax : +886-3-572-3694
ylin@cs.nthu.edu.tw

Abstract - We propose a software tool for automatic generation of hardware accelerators for performing Discrete Wavelet Transform (DWT) with user-specified coefficient parameters. In addition to (5, 3) and (9, 7) DWT filters adopted by the next generation JPEG2000 image compression standard, other useful filters such as (9, 3), (6, 10), and (2, 2) can also be generated. The generated hardware IPs can perform both forward and inverse transform (FDWT and IDWT). We analyze variable life time for register allocation with low power consumption and apply register retiming technology to improve circuit performance. Our tool also produces on-chip-bus interface circuit compliant with the AMBA protocol together with associated device driver so that the generated IPs is 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.[1]

I. Introduction

For multimedia processing, Discrete Wavelet Transform (DWT)-based image coding has better performance than traditional Discrete Cosine Transform (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 [16], Motion-JPEG2000 [17], and MPEG4 still image coding [18]. 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 coefficients. 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 in multimedia SOC design.

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 image data in larger scale to eliminate the annoying blocking artifacts suffered by traditional DCT-based image coding. DWT can be done in either lossy or lossless mode. If we choose appropriate FDWT/IDWT pairs, perfect reconstruction can be achieved. The most important property 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 previously proposed architectures. 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 optimized lifting-based DWT theory and its corresponding architecture. In Section V, we present our synthesis, DFT, and ATPG results. 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 [11]. Daubechies applies DWT to image coding and proposed many famous wavelet filters [6] including the (9, 7) filter. Swendens proposed the Lifting Scheme (LS) [5] making DWT more computationally efficient. Calderbank, Daubechies and Sweldens later proposed the Integer Wavelet Transform (IWT) [2] that is more efficient without scarifying the performance. Vishwanath proposed the Recursive Pyramid Algorithm (RPA) [14] and a systolic array implementation. Ferretti proposed a modified RPA [7] to solve the boundary problem encountered during perfect reconstruction.

[1]* Supported in part by the National Science Council, R.O.C., under Grants No. NSC 92-2218-E-007-023, NSC 92-2220-E-007-009, and NSC 92-2220-E-007-017, by the Ministry of Economics Affairs, R.O.C., under Grant No. 92-EC-17-A-03-S1-0002, and by Taiwan Semiconductor Manufacturing Corp. (TSMC) under Grant No. 93A0002EA.

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[8]. Fig. 1 depicts the DWT functionality of JPEG2000.

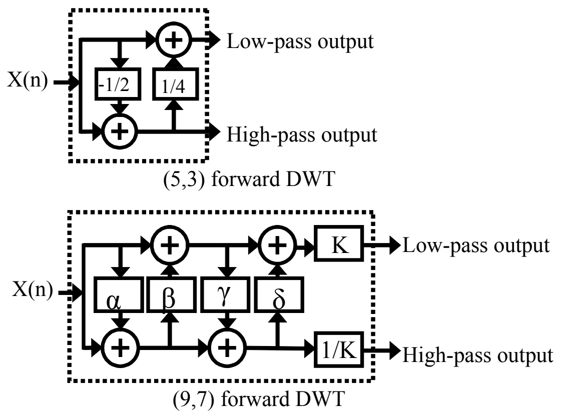


Fig. 1 The DWT functionality in JPEG2000

It is easy to see that the structure of a (9, 7) filter is a cascade of two (5, 3) filters. Chen et al [3] proposed a combined (5, 3) filter and (9, 7) filter architecture by means of folding as shown in Fig. 2.

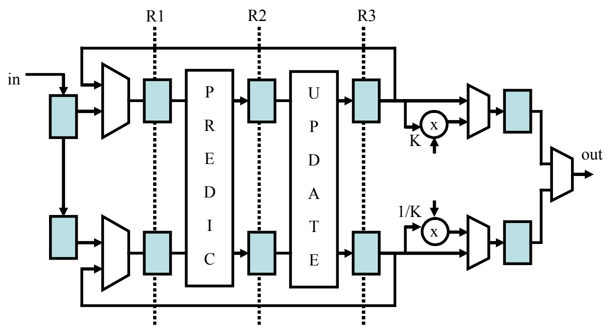


Fig. 2 Chen's Combined (5, 3) and (9, 7) Filter Architecture

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 of **Fig. 3**, we find that many components are common to both directions. The (9, 7) filter also exhibits similar characteristics as shown in Fig. 4.

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

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

(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).

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

(9,7) filter Forward Discrete Wavelet Transform

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

(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 Fig. 5, we can perform both FDWT and IDWT using the same architecture. Compared with the architecture proposed by Chen [3], our architecture needs only six additional multiplexers to gain the dual mode capability.

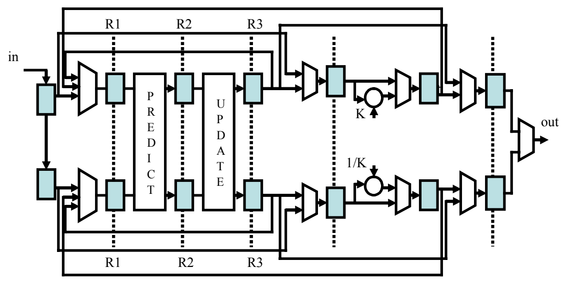


Fig. 5 Proposed Dual Mode FDWT/IDWT Architecture

B. *The bit precision*

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

In Reference [8], the PSNR (db) value of image Lena for different rate (bpp) and different fraction bit-length (m) is studied. TABLE 1 shows that using 8 bits for the fraction bits has great quality. In order to get higher quality we use 10 bits for the fraction part to approximate to the case of infinite bit-length.

TABLE 2
PSNR as a function of bit precision for (9, 7) filter

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.

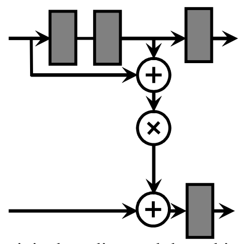
According to the allocation table, we design a finite state machine (FSM) to control the data path of nine registers. In our application, we need four FSMs to perform both FDWT and IDWT of both (5, 3) and (9, 7) filters.

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 uses shift operation and addition to implement 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 similarly.


The original predict module architecture

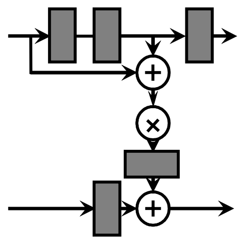

The new predict module architecture after retiming

Fig. 6 Optimizing the predict module by retiming

IV. Generalized DWT

In this section, we generalize our propose DWT architecture to filters with different coefficients. First we will review the lifting-based DWT theory proposed by Swelden. Then, we propose detail architecture of each sub-module. We try to reuse our dual mode FDWT/IDWT architecture for different factoring of DWT. Finally, we describe our parameterized DWT generator.

Given user input of the coefficients of wavelet filter taps, our parameterized DWT generator will produce both the DWT C model and the DWT RTL model (DWT IP). We feed the image data into both the DWT C model and the DWT IP. Comparing the outputs from both models helps us in functional verification.

A. *Lifting-based DWT theory*

Applying the Euclidean algorithm that finds the greatest common divisor of two natural numbers, we can find the greatest common divisor of two Laurent polynomials. We can factor any two Laurent polynomials $a(z)$ and $b(z)$ into the following form:

$$\begin{bmatrix} a(z) \\ b(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_n(z) \\ 0 \end{bmatrix} \tag{1}$$

Any wavelet filter can be represented as a polyphase matrix below:

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_0(z) & g_0(z) \end{bmatrix} \tag{2}$$

And we know that the perfect reconstruction property is

$$P(z)\widetilde{P}(z^{-1})^t = I \tag{3}$$

However, we can factor $h_e(z)$ and $h_o(z)$ into the form:

$$\begin{bmatrix} h_e(z) \\ h_0(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_n(z) \\ 0 \end{bmatrix} \tag{4}$$

And it is easy to build a complementary polyphase matrix:

$$P^0(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_0(z) & g_0(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} K & 0 \\ 1 & 1/K \end{bmatrix} \tag{5}$$

The original polyphase matrix can always be constructed from $P^0(z)$ with one lifting

$$P(z) = P^0(z) \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \tag{6}$$

Then observe that:

$$\begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & q_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ q_i(z) & 1 \end{bmatrix} \tag{7}$$

We know that any wavelet filter's polyphase matrix can be factored into the form

$$P(z) = \prod_{i=1}^{n} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \tag{8}$$

This is the most important result.

B. The factorization of different wavelet filters

We have already known that the division of two Laurent polynomials is not unique. Different divisions of polynomials result in different factoring of wavelet filters. We will analyze the factoring of many wavelet filters including both the odd taps wavelet filters and the even taps wavelet filters.

B.1 The factorization of odd taps wavelet filters

Focusing on the factoring of odd taps wavelet filters, let's analyze the polyphase matrix.

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_0(z) & g_0(z) \end{bmatrix} \tag{9}$$

Because the length of the wavelet filter is odd, the length of the even part transfer function $h_e(z)$ should be longer than the length of the odd part transfer function $h_o(z)$. And the interval between them is just one. Therefore we can fix the division. The quotient polynomials $q_i(z)$ must be in the form of either $\alpha(z+1)$ or $\alpha(1+z^{-1})$. To avoid the value α being too large or too infinitesimal, we put the value α under some constraints. If its value is neither larger than 0.000005 nor smaller than -0.000005, α will be truncated. By those principles, we can factor all odd taps wavelet filters into the lifting scheme. The results are shown in TABLE 1. In the table, both a_0 and a_1 are equal to α .

TABLE 1

The factorization results of the odd length wavelet filters

$s_i(z), t_i(z) = a_0 z^{d_M} + a_1 z^{d_{M-1}} + a_2 z^{d_{M-2}}$					
(9 , 7)					
	d_M	a_0	a_1	a_2	K
$s_1(z)$	0	0	0		
$t_1(z)$	0	-1.586146	-1.586146		
$s_2(z)$	1	-0.052979	-0.052979	0	0.869865
$t_2(z)$	0	0.882926	0.882926		
$s_3(z)$	1	0.443504	0.443504		
(5 , 3)					
	d_M	a_0	a_1	a_2	K
$s_1(z)$	0	0	0		
$t_1(z)$	0	-0.5	-0.5	0	1
$s_2(z)$	1	0.25	0.25		
(9 , 3)					
	d_M	a_0	a_1	a_2	K
$s_1(z)$	0	0	0		
$t_1(z)$	0	-0.499992	-0.499992	0	0.707110
$s_2(z)$	1	-0.046875	-0.046875		
(13 , 11)					
	d_M	a_0	a_1	a_2	K
$s_1(z)$	0	0	0		
$t_1(z)$	0	-2.254075	-2.254075		
$s_2(z)$	1	-0.190816	-0.190816		
$t_2(z)$	0	-0.196851	-0.196851	0	0.983757
$s_3(z)$	1	0.106976	0.106976		
$t_3(z)$	0	1.591717	1.591717		
$s_4(z)$	1	0.578168	0.578168		

B.2 The factorization of even taps wavelet filters

Now let's focus on the polyphase wavelet filters. The length of transfer function $h_e(z)$ and transfer function $h_o(z)$ are the same. If we still limit the quotient polynomials $q_i(z)$ in the form of $\alpha(z+1)$ or $\alpha(1+z^{-1})$, the factorization will fail.

TABLE 2

The factorization results of the odd length wavelet filters

$s_i(z), t_i(z) = a_0 z^{d_M} + a_1 z^{d_{M-1}} + a_2 z^{d_{M-2}}$					
(6 , 2)					
	d_M	a_0	a_1	a_2	K
$s_1(z)$	0	0	0	0	
$t_1(z)$	0	-1	0	-1	0.707107
$s_2(z)$	1	-0.0625	0.5	-0.0625	
(2 , 2)					
	d_M	a_0	a_1	a_2	K
$s_1(z)$	0	0	0		
$t_1(z)$	0	1	1	0	0.707107
$s_2(z)$	1	0.5	-0.5		
(10 , 6)					
	d_M	a_0	a_1	a_2	K
$s_1(z)$	0	0	0	0	
$t_1(z)$	0	0.369515	0	0	
$s_2(z)$	1	0.119532	0.119532	0	
$t_2(z)$	0	3.242383	3.242383	0	1.149596
$s_3(z)$	0	-0.008568	-0.008568	0	
$t_3(z)$	0	-4.611887	-4.611887	0	
$s_4(z)$	1	-0.110879	0.500002	0.110878	

We must choose the quotients so that the gcd is a constant. Therefore we use another adaptive factorization. If the lengths of two transfer functions are the same, we let the quotient be a constant α. If their difference is one, we use $\alpha(z+1)$ or $\alpha(1+z^{-1})$ to be the quotient. For both cases, we use normal division without limited quotient. In the last case, a_0, a_1 and a_2 may be different. By the above guideline, the quotients of the normal cases will be the alternate form of a constant α and $\alpha(z+1)$ or $\alpha(1+z^{-1})$. The gcd will be a constant as expected.

B.3 The factorization of inverse discrete wavelet transform

Another important issue is how to factor inverse wavelet transform. According to the theory by Mallat [5], each factored wavelet filter will be in the form shown in Fig. 7. The scaling part is K and 1/K for FDWT. But, it is 1/K and K for IDWT. The lifting part is $P^{(1)}, U^{(1)}, \ldots P^{(M)}, U^{(M)}$ for FDWT and $U^{(M)}, P^{(M)}, \ldots U^{(1)}, P^{(1)}$ for IDWT. The block diagrams of the FDWT and IDWT are symmetric. Therefore we only need to factor the FDWT. We can easily construct the inverse wavelet transform from the forward wavelet transform by means of lifting.

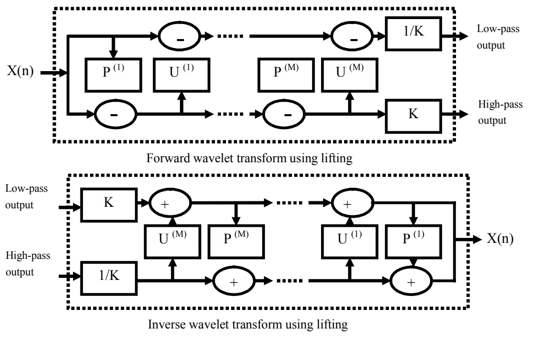


Fig. 7 Generalized FDWT and IDWT in Lifting Scheme

C. Generalized DWT Architecture

In the previous sub-section, we have already introduced the generalized DWT theory. Now, we present the architecture.

C.1 Top-Level DWT architecture

The top-level generalized FDWT/IDWT architecture is based on the dual mode DWT architecture we have proposed. The previous architecture already can perform (5, 3) and (9, 7) FDWT/IDWT operation. Now we try to make it more generalized for filters with different coefficients.

Our new generalized DWT architecture is different from the previous one that can only deal with two couples of lifting and the dual lifting. If some wavelet filter is factored into more than two couples of lifting, we must extend the architecture. The easiest way is to cascade more predict and update modules. We also have to give the K module new K and 1/K values. The OUT module remains the same.

C.2 Predict/Update module

Fig. 8 depicted a modified FDWT/IDWT architecture optimized for (5, 3) and (9, 7) filters employed by the JPEG2000 standard. From the factoring of (5, 3) and (9, 7) filters, we find that their quotient polynomials $q_i(z)$ are in the form of $\alpha(z+1)$ or $\alpha(1+z^{-1})$. Therefore, if we keep every quotient in our polynomial division as the form of $\alpha(z+1)$ or $\alpha(1+z^{-1})$, it is easy to map the lifting scheme into our FDWT/IDWT architecture.

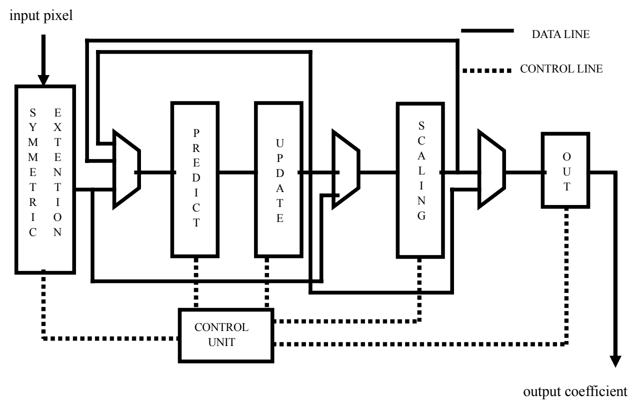


Fig. 8 A Modified FDWT / IDWT Architecture

Let’s focus on the PREDICT and UPDATE modules. Fig. 9 depicts the original PREDICT module. The only thing we need to do is to replace the multiplier α. We implement an automatic shift-add multiplier generator to produce a low-power and high-speed multiplier. The scaling module also uses the generated shift-add multiplier. Then the generalized shift register module is very easy to extend too. After we factor the polyphase matrix, if we need one lifting or one dual lifting, we use the (5, 3) mode. If we need two liftings or two dual liftings, we use the (9, 7) mode. Therefore, we can handle any factoring of wavelet filter.

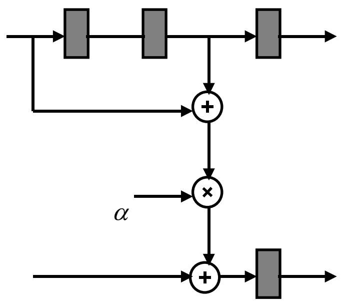


Fig. 9 Original PREDICT Module

According to Grangetto’s work [18], we know that a fixed quotient form may not be the optimal factoring of a wavelet filter. The quotient may be in the form of -------- resulting in a PREDICT module as depicted in Fig. 10. Therefore, we can construct any generalized FDWT/IDWT architecture with different wavelet coefficients.

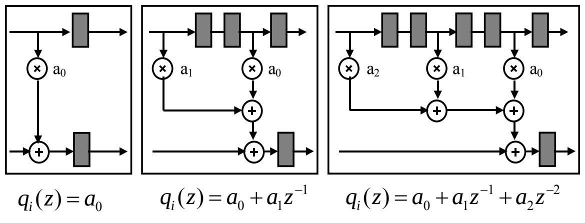

a0
a1
a0
a2
a1
a0
$q_i(z) = a_0$
$q_i(z) = a_0 + a_1 z^{-1}$
$q_i(z) = a_0 + a_1 z^{-1} + a_2 z^{-2}$

Fig. 10 Generalized PREDICT Module

C.3 Signal extension module

We have already proposed a low power signal extension module for both (5, 3) and (9, 7) filters. Now, a general architecture is needed. We have found that the signal extension module is related to the number of the couples of the lifting and the dual lifting. Therefore we pre-design the signal extension module for one, two, three and four couples of the lifting and the dual lifting. We select the signal extension module according to the number of couples of lifting.

C.4 Control unit module

The control unit of the top DWT architecture is very complex. It should control every sub-module and generate the output valid signal. Our DWT generator will calculate the latency of the signal extension module and the total latency of the top-level DWT module and produce a controller with appropriate timing behavior.

V. DWT IP Implementation

We have implemented the proposed architecture as a reusable IP generator. We follow the guidelines defined in the Reuse Methodology Manual (RMM).

TABLE 3

Parameter Set of the IP Generator

Parameter	Description	Parameter Rang
Direction	Forward DWT Or Inverse DWT	FDWT
		IDWT
Mode	Wavelet Filter Coefficients	Lossless (5,3) filter
		Lossy (9,7) filter
De-interleaved	Input / Output Ordering	De-interleaved (LLL…LHHH…H)
		Non-de-interleaved (LHLHLH…LH)
Length	Row (Column) Length	16 ~ 1K Pixels
Stride	Steps between Pixels	At least 1 pixel, at most one row

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 at 5. We need 152,371,200 clock cycles and the clock period bound is 3.281 ns.

Using the TSMC 0.13μm cell library, we are able to achieve this target. After synthesis, we perform DFT synthesis to insert scan chains. The report is given in Fig. 11. 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.

Timing constraint (insert DFT or not)		No constraint	4ns	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 0.13μm cell library >

Fig. 11 Synthesis and DFT report

VI. Verification and Integration

To verify our generated 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 integrated the DWT IP into an AMBA-AHB-based SOC platform as depicted in Fig. We use a data-driven architecture. The DWT IP is active only when data arrives.

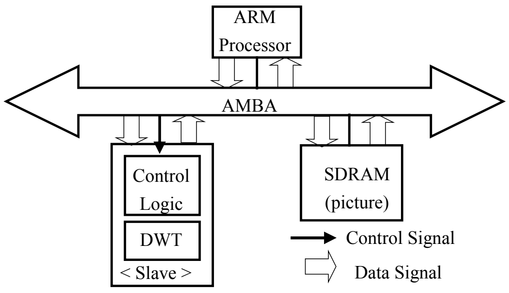


Fig. 12 The DWT IP AHB Interface

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

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

Finally, we integrate all software and hardware together in an SOC platform as depicted in Fig. 13. The Jasper software is running on the ARM922T CPU with the DWT function replaced by the function call to the DWT device driver.

Both the ARM922T CPU and the DWT IP in FPGA are clocked at 25MHz. When encoding a 512x512—pixel image, the pure software version takes 7.218 sec while the DWT-accelerated version takes 6.631 sec. This demonstrates the functional correctness of our proposed generator.

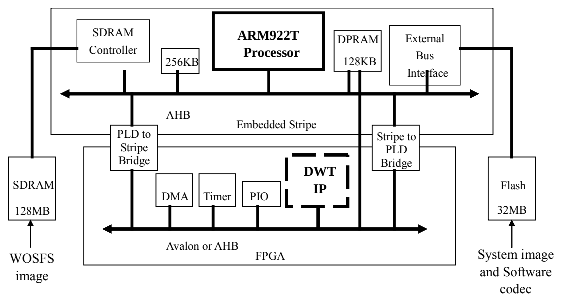


Fig. 13 SOC Integration for JPEG2000 Codec

VIII. Conclusions

We have proposed a software tool for automatic generation of hardware accelerators for performing Discrete Wavelet Transform (DWT) with user-specified coefficient parameters. In addition to (5, 3) and (9, 7) DWT filters adopted by the JPEG2000 image compression standard, other useful filters such as (9, 3), (6, 10) and (2, 2) filters can also be generated. Our generated hardware IP can perform both forward and inverse transform (FDWT and IDWT). We have employed from high level synthesis research the register life time analysis technology for reducing power consumption and logic retiming technology for improving circuit performance. Our tool also produces on-chip-bus interface circuit compliant with the AMBA protocol together with associated device driver so that the generated IPs is ready for SOC integration.

We have verified 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.

In the future, we can try to integrate the DWT architecture that we generated into other image or video compression system. Furthermore, we can improve the data communication bottleneck between the DWT IP and the memory modules.

References

[1] 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.

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

[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. Chrysafis and A. Ortega, “Line-based, reduced memory, wavelet image compression,” IEEE

Transactions on Image Processing, Vol. 9, No. 3, pp. 378-389, March 2000.

[5] A. Cohen, I. Daubechies and J. Feauveau, "Bi-orthogonal bases of compactly supported wavelets", Comm. Pure Appl. Math., Vol. 45, pp. 485-560, 1992.

[6] I. Daubechies and W. Swelden, "Factoring wavelet transform into lifting steps," The Journal of Fourier Analysis and Applications, Vol. 4, pp.247-269, 1998.

[7] M. Ferretti and D. Rizzo, ``Handling borders in systolic architectures for the 1-D discrete wavelet transform for perfect reconstruction,'' IEEE Transactions on Signal Processing, Vol. 48 , Issue 5 , May 2000, pp. 1365 - 1378

[8] M. Grangetto, E. Magli, M. Martina, and G. Mo, "Optimization and Implementation of the Integer Wavelet Transform for Image Coding," IEEE Transactions on Image Processing, Vol. 11, Issue 6, pp. 596-604, June 2002.

[9] 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.

[10] 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, Vol.2, pp. 445 – 448, May 2001.

[11] S. Mallat, "A theory for multi resolution signal decomposition: the wavelet representation," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, pp. 674-693, July 1989.

[12] 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, August 2002.

[13] 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, August 2003.

[14] M. Vishwanath, "The Recursive Pyramid Algorithm for the Discrete Wavelet Transform," IEEE Trans. on Signal Processing, Vol. 42, No. 3, pp. 673-676, March 1994.

[15] 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.

[16] ISO/IEC. ISO/IEC 15444-1. Information technology – JPEG2000 Part I. image coding system. March 2000.

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

[18] ISO/IEC. JTC1/SC29/WG11 Coding of moving pictures and audio, January 2001.