HARDWARE IMPLEMENTATION OF SHAPE-ADAPTIVE DISCRETE WAVELET TRANSFORM WITH THE JPEG2000 DEFAULTED (9,7) FILTER BANK

Chao-Tsung Huang, Po-Chih Tseng, and Liang-Gee Chen

DSP/IC Design Lab, Graduate Institute of Electronics Engineering, and
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.
Email:{cthuang, pctseng, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

In this paper, an efficient hardware implementation of two-dimensional shape-adaptive discrete wavelet transform (2-D SA-DWT) with the JPEG2000 defaulted (9,7) filter bank is presented. Two techniques are used to minimize the critical path and the internal buffer size. One technique is the flipping structure which can shorten the critical path of the lifting scheme by flipping multiplier coefficients, rather than pipelining. The other technique is a shape-adaptive boundary handling strategy which can enhance lifting-based architectures to solve boundary extension problems of SA-DWT with little hardware overhead. A prototyping chip of this implementation will be fabricated with TSMC 0.25μm CMOS 1P5M process, and the estimated frequency and the core area are 50 MHz and 2.83 mm^2, respectively.

1. INTRODUCTION

Visual object coding has become an important technique because it can provide great flexibility to manipulate visual objects in multimedia applications and could improve visual quality in very low bit-rate applications. There have been many research efforts on developing new algorithms for coding arbitrarily shaped visual objects. SA-DWT [1] has been proven to outperform other coding methods in both PSNR and subjective quality. The emerging visual coding standard, MPEG-4, has adopted SA-DWT as its core transform for coding arbitrarily shaped still texture.

Since the Daubechies (9,7) filter bank could achieve better performance than other filter banks [2], the emerging still image coding standard, JPEG2000, has adopted it as the defaulted lossy DWT filter bank. The DWT architectures can be implemented by the convolution-based and the lifting-based methods. However, the boundary extension issue is not frequently discussed in literature, which could result in modification of the original architectures. The implementation of SA-DWT relies on the ability to handle the boundary extension for any kind of input signal segments. In this paper, the symmetric extension is considered, instead of zero padding and periodic extension because of its good performance [3].

In [4] and [5], symmetric boundary extension issues have been considered for convolution-based architectures, while only [6] mentioned about the boundary handling strategy for lifting-based architectures. However, these boundary extension strategies would require more than the minimal number of registers in lifting-based architectures. The number of registers will dominate the hardware resource of 2-D line-based DWT architectures because it is proportional to the internal buffer size [7].

In this paper, the 1-level 2-D line-based architecture [7] is adopted for reducing the external memory access and achieve 100% hardware utilization. Besides, the method of [6] is modified to maintain the minimal number of registers in lifting-based architectures and extended to the flipping structure [8]. The organization of this paper is as follows. The SA-DWT algorithm and DWT architectures are reviewed in section 2 and 3, respectively. Section 4 presents the proposed shape-adaptive boundary handling strategy. The comparison results are given in section 5, and the prototyping chip implementation is shown in section 6. Finally, conclusions about this paper are provided in section 7.

2. SA-DWT ALGORITHM

Visual objects mainly consist of the shape and texture information. The former illustrates which positions belong to the object, and the latter describes the texture content. The basic concept of the 1-D SA-DWT algorithm is to treat the arbitrarily shaped visual object as several continuous signal segments based on the shape information and to perform DWT on the texture information for each signal segment independently. This concept can be directly extended to the 2-D SA-DWT for separable 2-D DWT filter banks. Without loss of generality, the row-wise DWT is assumed to be performed before the column-wise DWT for the separable 2-D DWT. There are two more issues that should be considered for SA-DWT. The first one is about the local and global subsampling strategies, and the other one is the boundary extension issue of very short signal segments.

The local subsampling strategy selects the subsampling positions that refer to positions relative to the beginning of each signal segment. Thus, the number of lowpass signals is always more than or equal to that of highpass signals, and this is preferred for the entropy coding methods. The global subsampling strategy will refer to positions relative to the rectangular box of the shape information. Although the highpass signals may be more than the lowpass signals, the phase of row-wise SA-DWT coefficients can be always aligned for the column-wise SA-DWT. According to [1], the global subsampling can achieve better coding gain than the local subsampling. We will only focus on the global subsampling in the following discussion.

0-7803-7750-8/03/$17.00 ©2003 IEEE

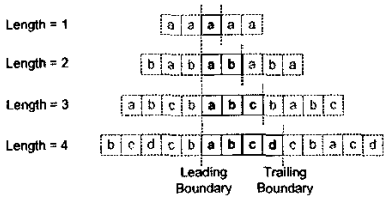

Fig. 1. Symmetric boundary extension for very short signal segments

The other issue is how to handle the boundary extension for very short signal segments. As described in [1], the leading boundary extension is performed first and then the trailing boundary. Some extension examples are given in Fig. 1, where the solid and dot rectangles represent the signal segments and the extension signals, respectively. The boundary extension of very short signal segments is more complex than that of long signal segments in which the leading and trailing boundary extensions are independent.

3. DWT ARCHITECTURES

In this section, we will categorize the DWT architectures and discuss the implementation issues for boundary extension. The 1-D DWT architectures focus on the computing unit designs, including multipliers, adders, and registers. The 2-D DWT architectures are dominated by memory issues, such as frame memory access bandwidth and internal buffer size.

3.1. 1-D DWT Architectures

The 1-D DWT architectures are mainly convolution-based or lifting-based. The convolution-based ones can be implemented with serial or parallel filters [9]. The parallel filters can handle the boundary extension with a router which is between the registers and input signals [4, 5]. The router is basically a multiplexer and will be very complex if it is designed for solving the boundary extension of very short signals. Unlike parallel filters, serial filters can not handle the boundary extension without additional registers because of the serial feature of input signals.

The lifting-based architectures are superior to the convolution-based ones in the number of multipliers, adders, and registers. For example, the parallel filter implementation of the JPEG2000 defaulted (9,7) filter bank would require 9 multipliers, 14 adders, and 7 registers. And the lifting-based implementation can only requires 6 multipliers, 8 adders, and 6 registers, as shown in Fig. 2(a) where the grey nodes represent registers, and the coefficients are given as $a = -1.586134342$, $b = -0.052980118$, $c = 0.882911076$, $d = 0.443506852$, and $K = 1.149604398$.

However, the critical path of Fig. 2(a), $4T_a + 8T_m$ is much longer than the parallel filter implementation, $T_a + 4T_m$, where T_a is the time needed for an addition operation, and T_m is the time taken for a multiplication operation. Although pipelining the lifting-based architecture can reduce the critical path, it would also increase the number of registers. For example, if Fig. 2(a) are cut with 4 pipelining stages so as to reduce the critical path to $T_a + 2T_m$, it would increase by 6 registers. We have proposed the flipping

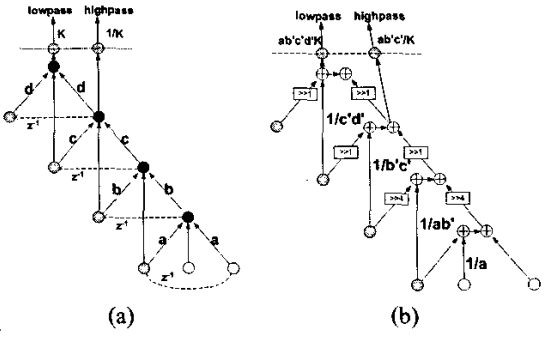

Fig. 2. (a) Lifting structure for (9,7) filter; (b) Flipping structure for (9,7) filter

structure to shorten the critical path to $T_a + 5T_m$ by flipping the multiplier coefficients without any hardware overhead, as shown in Fig. 2(b) where $b' = 16b$, $c' = 2c$, and $d' = 2d$ [8].

For software implementation, the boundary extension of this lifting structure can be solved as described in [10]. As for hardware implementation, we have proposed a shape-adaptive boundary extension strategy in [6], where two additional registers are used to address the special case that the signal segment length is one.

3.2. 2-D DWT Architectures

According to the external frame memory access bandwidth and the internal buffer size, the 2-D DWT architectures can be categorized as direct, 1-level 2-D, and multi-level 2-D [7]. The direct method performs the row-wise DWT first, and then the column-wise DWT with one 1-D DWT architecture. This direct implementation is very simple but requires huge external frame memory bandwidth. The 1-level 2-D architecture performs the row-wise and column-wise DWT of the same level simultaneously. The implementation method is to use some internal buffer to store temporal coefficients so as to reduce the external frame memory access. The multi-level 1-D architecture performs all levels of 2-D DWT at the same time and makes the required external memory bandwidth minimized. However, this kind of implementation usually results in a poor hardware utilization. For example, using Recursive Pyramid Algorithm to schedule the decomposition tasks will make the hardware utilization only 66.7% [11]. These three architectures are summarized in Table 1, where the decomposition level, J, is assumed to be infinite, N is the image width, and L is a constant related to the adopted 1-D DWT architecture. For the detailed description, please reference to [7].

For the (9,7) filter bank, the L of the parallel-parallel architecture is 8.5 [9]. However, for the non-pipelining lifting-based architecture, the L can be only 5.5 [7].

4. SHAPE-ADAPTIVE BOUNDARY HANDLING

Except the normalization step, Fig. 2(b) is composed of four basic flipping units, and each unit consists of one multiplier and two adders. For solving the boundary extension issue, we propose to modify each basic flipping unit to the Shape-Adaptive Boundary

Architecture	External Memory Access	Internal Buffer Size	Hardware Utilization
Direct	$5.33N^2$	No	100%
1-level	$2.67N^2$	LN	100%
Multi-level	$2N^2$	2LN	Low

Table 1. Summary of 2-D architectures ($J \rightarrow \infty$)

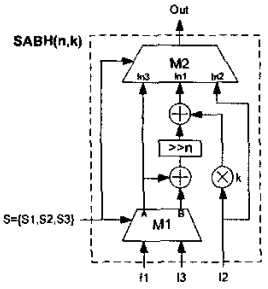

Fig. 3. Shape-adaptive boundary handling (SABH) for the flipping structure unit

Handling (SABH) unit as shown in Fig. 3, where n and k are the shift bit number and the multiplier coefficient, respectively. The two multiplexers, M1 and M2, can help the flipping unit to handle the boundary extension with examining the shape information $S = \{S1, S2, S3\}$ which is corresponding to the input signals $\{I1, I2, I3\}$. If the input signals are all inside the signal segment, M1 will set the nodes A and B as I1 and I3. Otherwise, when the input signals are at the segment boundary, M1 will set A and B both as I1 or I3, which depends on that the boundary is leading or trailing. And M2 will output the computation result In1 in the above conditions.

The above strategy is very similar with [6]. However, the strategy for the special case, the signal length is only one, is quite different in this paper. We propose to pass the signal of length one, which may be from even or odd positions, through the registers and the multiplexers, M1 and M2. Since this special case is undefined in the above conditions, the multiplexers can be designed to pass the signal to the lowpass node in the exact cycle. Thus, M2 will pass In2 or In3 in the special case.

By adopting SABH units to Fig. 2(b), the shape-adaptive flipping structure can be derived as Fig. 4, where the critical path is only increased by $3T_a$ if the time taken for multiplexers is ignored. The additional multiplexer, M_L, is used to select the correct lowpass signals and the shape information of lowpass and highpass signals with examining the shape information, $\{m2, m3, m4, m5\}$. Moreover, the right part of the dot line can be implemented independently from the left part. When Fig. 4 is extended to the 2-D line-based architecture, only the four data registers and four shape registers of the left side are required to be modified to the the internal temporal buffer because the right side can be implemented with registers independently.

Architecture	Multiplier	Adder	Critical Path	Internal Buffer Size (words)	Shape-Adaptive Handling
Parallel-Parallel	18	28	Tm+4Ta	8.5N	No
Previous Lifting	16	16	4Tm+8Ta	9N	Yes
Proposed	14	16	Tm+8Ta	5.5N	Yes

Table 2. Comparisons of 2-D DWT architectures for the (9,7) filter

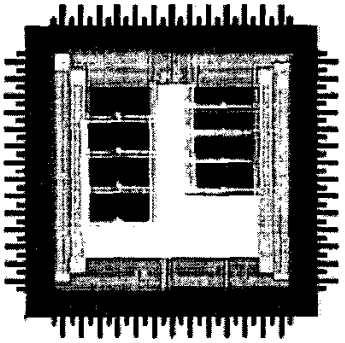

Fig. 5. Layout of the prototyping chip

5. COMPARISON

This section presents the comparison of three 1-level 2-D architectures, including parallel-parallel [4], previous lifting-based [6], and the proposed flipping architectures. The comparison results are given in Table 2, where the time taken for multiplexers is ignored for calculating the critical path. The parallel-parallel architecture requires more multipliers, adders, and internal buffer, but has a shorter critical path. However, the router design in [4] only can handle the boundary extension of long signal segments and will be very complex for handling very short signal segments. Although the previous lifting-based architecture can handle the shape-adaptive boundary extension, the critical path is too long, and the internal buffer size is larger than the parallel-parallel one. By adopting the proposed SABH units to the flipping structure and a proper design of the data buffer [7], the critical path is shortened, and the internal buffer size is only about 65% of the parallel-parallel one.

6. CHIP IMPLEMENTATION

A prototyping chip for the 1-level 2-D line-based SA-DWT with the (9,7) filter by using the proposed shape-adaptive flipping structure has been implemented and will be fabricated with TSMC 0.25-μm CMOS 1P5M process. The layout and chip feature are shown in Fig. 5 and Table 3, respectively. If this chip works at 50 MHz, the processing capability will be 100M pixels per second. This processing rate can afford the real-time computation of 1-level 2-D SA-DWT decomposition for the HDTV image size (1920 × 1088, YUV 4:2:0) at 30 frames per second since:

$$1920 \times 1088 \times 30 \times 1.5 \simeq 94 \times 10^6$$

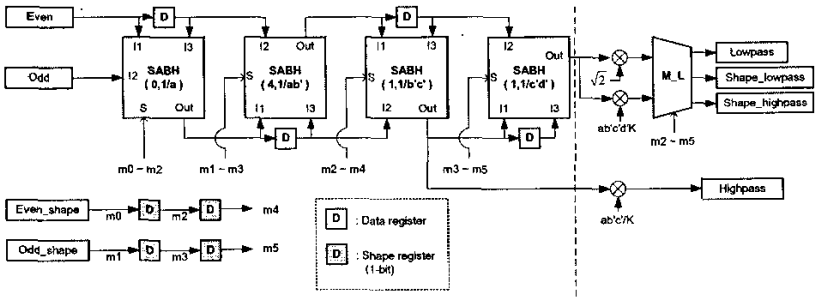

Fig. 4. Shape-adaptive flipping structure for the (9,7) filter

Technology	TSMC 0.25um CMOS 1P5M
Die Size	3.247 x 3.241 mm^2
Core Size	1.683 x 1.679 mm^2
Logic Gate Count	29551
On-Chip Memory	4 128x17 two port RAMs for temporal buffer
	2 64x17 two port RAMs for data buffer
	2 32x17 two port RAMs for data buffer
	Total 11968 bits
Transistor Count	336552
Max Clock Rate	50MHz
Function	Shape-Adaptive 1-level 2-D DWT with JPEG2000 lossy (9,7) filter
Frame Size	128 x 128

Table 3. Chip feature

In this prototyping chip, the data wordlength and the frame size are assumed to be 16-bits and 128×128, respectively. Thus, the internal buffer size is $5.5 \times 128 \times (16+1) = 11968$ bits. Under these conditions, the logic part and the internal buffer cost nearly the same area. Therefore, the internal buffer will dominate the area cost if the frame width is larger than 128 or the data wordlength is longer than 16-bits.

7. CONCLUSION

In this paper, an efficient implementation of SA-DWT with the (9,7) filter is presented, in which the flipping structure and the shape-adaptive boundary handling unit are adopted. The former can shorten the critical path of the lifting-based architecture without additional hardware resource. And the latter can solve the shape-adaptive boundary extension issues with few multiplexers and no additional registers. When these two techniques are used for the 2-D line-based architecture, the internal buffer size can be minimized with a short critical path. The prototyping chip implementation can prove the efficiency in the processing capability and the internal buffer size.

8. REFERENCES

[1] S. Li and W. Li, "Shape-adaptive discrete wavelet transforms for arbitrarily shaped visual object coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 5, pp. 725–743, Aug. 2000.

[2] P. Mathieu M. Antonini, M. Barlaud and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[3] S. Mallat, *A wavelet tour of signal processing*, Academic Press, 1998.

[4] C. Chakrabarti, "A DWT-based encoder architecture for symmetrically extended images," in *IEEE International Symposium on Circuits and Systems*, 1999, vol. 4, pp. 123–126.

[5] K. Seth and S. Srinivasan, "VLSI implementation of 2-D DWT/IDWT cores using 9/7-tap filter banks based on the non-expansive symmetric extension scheme," in *15th International Conference on VLSI Design*, 2002, pp. 435–440.

[6] P.-C. Tseng, C.-T. Huang, and L.-G. Chen, "VLSI implementation of shape-adaptive discrete wavelet transform," in *Proc. of SPIE International Conference on Visual Communications and Image Processing*, 2002, pp. 655–666.

[7] P.-C. Tseng, C.-T. Huang, and L.-G. Chen, "Generic RAM-based architecture for two-dimensional discrete wavelet transform with line-based method," in *Asia-Pacific Conference on Circuits and Systems*, 2002, pp. 363–366.

[8] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Flipping structure: An efficient VLSI architecture for lifting-based discrete wavelet transform," in *Asia-Pacific Conference on Circuits and Systems*, 2002, pp. 383–388.

[9] C. Chakrabarti, M. Vishwanath, and R. M. Owens, "Architectures for wavelet transforms: A survey," *Journal of VLSI Signal Processing*, vol. 14, pp. 171–192, 1996.

[10] G. Xing, J. Li, S. Li, and Y.-Q. Zhang, "Arbitrarily shaped video-object coding by wavelet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 10, pp. 1135–1139, Oct. 2001.

[11] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," *IEEE Transactions on Signal Processing*, vol. 42, no. 3, pp. 673–677, Mar. 1994.