FLIPPING STRUCTURE: AN EFFICIENT VLSI ARCHITECTURE FOR LIFTING-BASED DISCRETE WAVELET TRANSFORM

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

DSP/IC Design Lab, Graduate Institute of Electronics Engineering,
National Taiwan University, Taipei, Taiwan, R.O.C.
Phone:+886-2-2363-5251, Fax:+886-2-2363-8247, Email:{cthuang, pctseng, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

Using lifting scheme to construct VLSI architectures of discrete wavelet transform outperforms using convolution in many aspects, such as computation complexity and boundary extension. Nevertheless, the critical path of lifting scheme is potentially longer than that of convolution. Although pipelining can reduce the critical path, it will prolong the latency and require more registers for 1-D architecture as well as larger memory size for 2-D line-based architecture. In this paper, an efficient VLSI architecture is proposed to provide a variety of hardware implementations for improving and possibly minimizing the critical path and memory requirement of lifting-based discrete wavelet transform by flipping conventional lifting structures. By case studies of JPEG2000 defaulted (9,7) filter and an integer (9,7) filter, the efficiency of the proposed flipping structure is shown.

1. INTRODUCTION

Since the discrete wavelet transform (DWT) was deduced by Mallat [1], many researches on wavelet-based signal analysis and compression have derived fruitful results due to its well time-frequency decomposition. Furthermore, emerging image coding standards, such as JPEG2000 still image coding and MPEG-4 still texture coding, have adopted the DWT as their transform coder. However, the DWT requires more arithmetic operations for its filter computation than the previous generation transforms such as the discrete cosine transform (DCT). Moreover, contrary to the block-based DCT, the DWT is basically frame-based. The huge amount of memory requirement and access bandwidth becomes the bottleneck of the implementation for 2-D DWT [2].

From the appearance of lifting scheme [3] and a factorization method of lifting steps [4], lifting scheme is widely used to reduce the computation of DWT as well as the control complexity of boundary extension and memory access. On the other hand, line-based method is proposed to shrink the internal memory requirement from frame size to a few line buffers by proper memory management [5]. Nonetheless, the trade-off between the critical path and the size of line buffer has not been discussed clearly in the literature. That is, as more pipelining stages are cut to meet shorter critical path, the required line buffer size will increase very rapidly. In this paper, we propose an efficient VLSI architecture for lifting-based DWT that can provide a variety of hardware implementations to improve and possibly minimize the critical path and the size of line buffer by flipping conventional lifting structures. In section 2, the concepts, conventional architectures, and problems of lifting-based, convolution-based, and 2-D DWT are described. Section 3 presents the proposed flipping structure while two case studies of (9,7) filter are given to prove the efficiency in section 4. Finally, conclusions about this paper are provided in section 5.

2. DWT ARCHITECTURE

There have been many VLSI architectures that are proposed for DWT implementation. Convolution-based and lifting-based architectures are the mainstream for 1-D DWT while level-by-level and line-based methods are the most feasible implementation for 2-D DWT. In the following, we will briefly describe them and discuss what problems may happen. For simplicity and comparison, only the case that throughout is 2-input/2-output per clock cycle is considered.

2.1. Convolution-based

The arithmetic computation of DWT is basically

0-7803-7690-0/02/$17.00 ©2002 IEEE

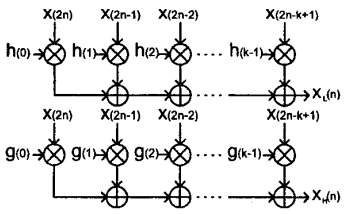

Figure 1: Simplest convolution-based architecture

filter convolution and downsampling as follows:

$$x_L(n) = \sum_{i=0}^{K-1} h(i) \cdot x(2n-i)$$
$$x_H(n) = \sum_{i=0}^{K-1} g(i) \cdot x(2n-i) \quad (1)$$

where x_L is the lowpass signal and x_H is the highpass signal.

Although [6] and [7] have proposed some efficient architectures for convolution-based DWT, only the simplest architecture shown in Fig. 1 can be used if throughout is set 2-input/2-output per clock cycle as well as the minimal latency and registers are required. In Fig. 1, the critical path is $T_m + (K-1) \cdot T_a$ where T_m is the delay time of a multiplier, T_a is the delay time of an adder, and K is the filter length while the hardware cost is $2K \cdot C_m + 2(K-1) \cdot C_a$ where C_m and C_a are the hardware cost of a multiplier and an adder, respectively. Furthermore, using adder tree can improve the critical path to $T_m + \lceil \log_2(K) \rceil \cdot T_a$. If linear DWT filter is adopted, we can save the required number of multipliers into the half by utilization of symmetric and anti-symmetric structures without modifying the critical path.

2.2. Lifting scheme

Lifting scheme is a new method for constructing wavelets by spatial approach [3]. Using lifting scheme has many advantages, such as fewer arithmetic operations, in-place implementation, and easily to manage boundary extension. According to [4], any DWT filter bank of perfect reconstruction can be decomposed into a finite sequence of lifting steps. This decomposition corresponds to a factorization of the polyphase matrix of target wavelet filter into a sequence of alternating upper and lower triangular matrices and a constant diagonal matrix as follows:

$$h(z) = h_e(z^2) + z^{-1} h_o(z^2)$$
$$g(z) = g_e(z^2) + z^{-1} g_o(z^2)$$
$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix} = \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \quad (2)$$

where $h(z)$ and $g(z)$ are the lowpass and highpass analysis filters respectively as well as $P(z)$ is the polyphase matrix.

Although the arithmetic gain of lifting scheme over convolution-based structures may possibly reach a factor four [8], the non-uniqueness of lifting factorizations diversifies the design space of hardware implementation for lifting-based DWT. In [9], a systematic design method for efficient VLSI architectures of lifting scheme is proposed, which includes specific lifting factorization, dependence graph formation, and systolic arrays mapping. The efficiency of lifting scheme with respect to hardware cost and the complexity of control circuits has been proven. However, the potentially long critical path is not discussed in detail and solved properly.

In order to give a brief explanation of this crisis, the popular (9,7) filter is taken as an example without loss of generality. We can decompose the (9,7) filter into four lifting stages and derive the signal flow graph as Fig. 2, where summation is performed at computing nodes. It is clear that the normalization step K and $1/K$ can be implemented independently or together with quantization if data compression is performed. Thus, we absorbedly focus on the implementation issue of the lifting stages. In comparison with the convolution-based architecture that requires 9 multipliers, 14 adders and 7 registers if adder tree and symmetry are adopted, Fig. 2 only needs 4 multipliers, 8 adders and 4 registers (K and $1/K$ are not involved). Nonetheless, the critical path of the lifting-based architecture is $4T_m + 8T_a$ while the convolution-based one merely costs $T_m + 4T_a$. Certainly, pipelining the lifting-based architecture can improve the serious timing problem but this will raise the number of registers rapidly. For instance, the lifting architecture for (9,7) filter can be pipelined to obtain a critical path $T_m + 2T_a$ with 6 additional registers. The penalty of pipelining lifting scheme will become very critical in the implementation of line-based 2-D DWT because the number of registers in 1-D DWT can dominate the hardware design of line-based 2-D DWT, which will be described later.

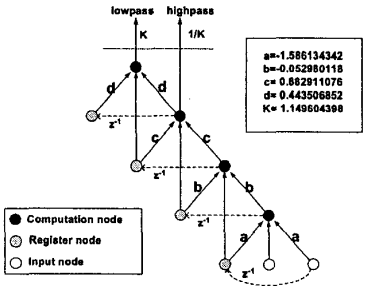

Figure 2: SFG for (9,7) filter

2.3. 2-D DWT

For hardware implementation of 2-D DWT, RAM-based 2-D DWT architectures possess many advantages, including real-life feasibility, high regularity/ density of storage and simple control circuits, than other architectures. According to the evaluation of [2], memory issue is the most important part for 2-D DWT implementation, unlike the 1-D case, in which the number of multipliers dominates. Generally speaking, we can put RAM-based architectures into two categories, level-by-level and line-based method. Which one is more preferred depends on what kind of hardware constraint is set. The external memory access, which consumes the most power and becomes very sensitive in the case of system-on-a-chip, of level-by-level method is more than double as much as that of line-based method. However, line-based architectures suffer the internal memory requirement.

Basically, the internal memory of line-based architectures consists of data buffer and temporal buffer as described in [10]. Furthermore, the data buffer is related to the input nodes of the signal flow graph for 1-D DWT and the size of temporal buffer is proportional to the number of register nodes. That is, the size of data buffer is fixed for any kind of architectures if throughput is the same while the temporal buffer is design-dependent. Thus, under the same timing criteria, how to minimize the temporal buffer will be the first consideration for hardware implementation of line-based DWT.

3. PROPOSED FLIPPING STRUCTURE

As the mentioned above, conventional lifting architectures cost fewer arithmetic operations but suffer the longer critical path than convolution-based architectures. In this section, we will describe the bottleneck of this timing problem and propose a VLSI architecture called flipping structure to solve it.

By observing (2), it can be easily found that lifting scheme is essentially composed of a series of computing stages. After the proposed lifting factorization of [9], we can derive an efficient lifting architecture that is a serial combination of computing units like Fig. 3(a). And the critical path is the sum of the timing delay for each computing units without pipelining. The accumulative effect results from that output of the computing node in each computing unit is the input node of the next computing unit. For reducing the timing accumulation, we suggest to eliminate the multiplier on the path from input node to computing node by flipping each computing unit with the inverse of the multiplier coefficient such as Fig. 3(b). Then every computing node can be splitted into two adders, one can process in parallel with other computing units and the other one is on the accumulative path, like Fig. 3(c). So, the timing accumulation can be heavily reduced by flipping the original lifting architectures. Another advantage of flipping structures is that no additional multipliers or adders will be required if the computing units are all flipped. Furthermore, we can put the flipped coefficients into the normalization step for assurance that correct lowpass and highpass coefficients will be obtained.

In fact, flipping structures can have many alternatives. How many computing units should be flipped is case-by-case and dependent on hardware constraints. Surely, the flipping method can be applied for lifting-based Inverse-DWT as well because the basic computing unit is the same as that of lifting-based DWT.

4. CASE STUDY

For showing the feasibility and efficiency of the proposed flipping structure, two case studies will be given in this section, including JPEG2000 default lossy DWT (9,7) filter and an integer (9,7) filter.

4.1. JPEG2000 (9,7) filter

Since the lifting structure of (9,7) filter is very regular and typical, it is meaningful to compare its conventional lifting architecture and flipping structure. The lifting architecture only needs 4 multipliers, 8 adders, and 4 registers with a critical path $4T_m + 8T_a$. The critical path can be reduced to $T_m + 2T_a$ by cutting 4 pipelining stages with 6 additional registers. Moreover, totally 34 registers can be used to minimize the critical path to a multiplier delay.

While flipping the lifting structure, the precision issue should be taken into consideration because there exist

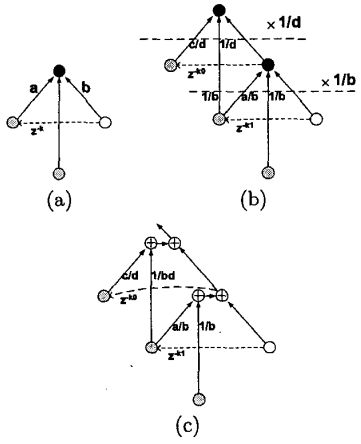

Figure 3: (a) basic computing unit; (b) flipping computing units; (c) after splitting computing nodes

the inverses of lifting coefficients. We design a feasible flipping structure for (9,7) filter as Fig. 4(a) by flipping all computing units, and the precision is carefully preserved for all coefficients. The critical path of this flipping structure is only $T_m + 5T_a$ without any additional hardware cost. Furthermore, if 3 pipelining stages are cut and some arithmetic operators are rearranged as Fig. 4(b), the critical path will be down to $T_m + T_a$ with 3 additional registers. The critical path can also be minimized to a multiplier delay by using 5 pipelining stages, such as Fig. 4(c), which merely needs 11 registers.

For obtaining more realistic results, we verify the above architectures with 12-bit precision for multiplier coefficients and 16-bit precision for intermediate data by using Verilog and then adopt Synopsys Design Compiler to synthesize circuits with standard cells from Avant! 0.35-μm cell library. The timing constraint is set as tight as possible for comparison. As a result, the synthesis results for critical path are very close to our prediction. At last, the experimental results are summarized in Table 1, where the case of convolution-based architecture is also listed.

4.2. Integer (9,7) filter

Besides the advantages of implementation issues, lifting scheme also can provide a well-constructed architecture to design wavelet-like filters. At the same time, both [11] and [12] propose a class of integer DWT filters based on the lifting scheme of (9,7) filter. Moreover, the

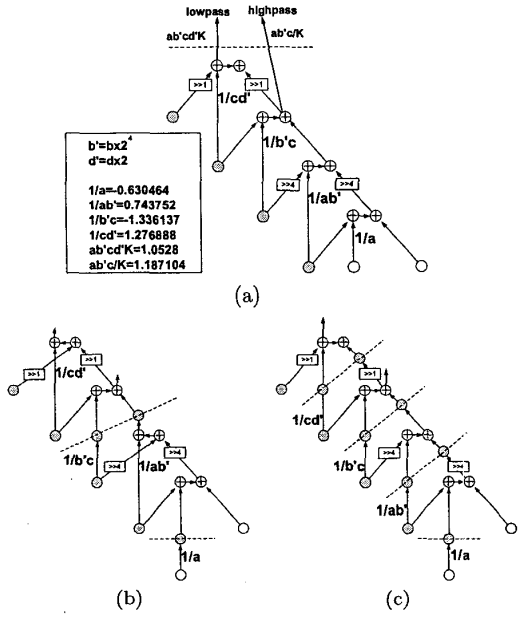

Figure 4: (a) flipping structure for (9,7) filter; (b) cut 3 pipeline stages; (c) cut 5 pipeline stages

	Multiplier	Adder	Critical path	Temp. buffer	Timing(ns)	Gate count
Conventional + no pipe.	4	8	4Tm+8Ta	4L	60	10084
Conventional + 4 stages	4	8	Tm+2Ta	10L	16	13082
Conventional + fully pipe.	4	8	Tm	34L	-	-
Flipping + no pipe.	4	8	Tm+5Ta	4L	21	10118
Flipping + 3 stages	4	8	Tm+1Ta	7L	12.6	11096
Flipping + 5 stages	4	8	Tm	11L	10	11244
Convolution + no pipe.	9	14	Tm+4Ta	7L	-	-
Convolution + 3 stages	9	14	Tm	23L	-	

Table 1: Comparison of several architectures for (9,7) filter

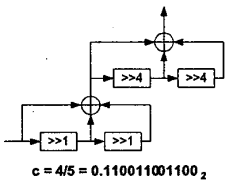

Figure 5: 12-bit precision multiplication of 4/5

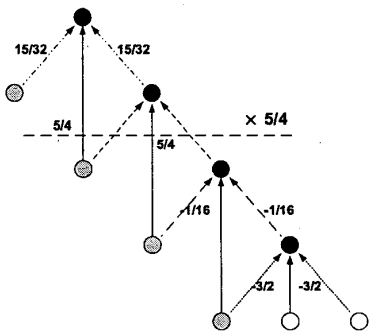

Figure 6: flipping structure of integer (9,7) filter

results of these two papers are the same and give a wonderful integer DWT filter with coefficients, $a = -3/2$, $b = -1/16$, $c = 4/5$, $d = 15/32$, and $K = 4\sqrt{2}/5$. This integer wavelet filter can reach nearly the same performance as the original (9,7) filter but requires much less hardware cost. We can use a few shifters and 2 adders to implement the multiplication of a, b, and d. But c needs a floating-point operation. If 12-bit precision of c is considered, only 4 adders and 4 shifters are sufficient for the implementation of multiplier c, such as Fig. 5. Therefore, if conventional lifting architecture is used for this integer DWT filter, 14 adders and some shifters can complete the hardware implementation with critical path $14T_a$.
However, if the computing unit of c is flipped as Fig. 6, 13 adders are required for the hardware implementation with critical path $7T_a$ by some modification of computing nodes. Besides, there is no floating-point operation in Fig. 6 and the precision of intermediate data can be reduced.

5. CONCLUSION

In this paper, we have proposed an efficient VLSI architecture for lifting-based DWT by flipping the conventional lifting architectures. Flipping structures can solve the serious timing problem of lifting scheme and reduce the size of temporal buffer for line-based DWT to the extent of minimum possibly. It also can provide a variety of well-featured architectures for any kind of DWT filters.

REFERENCES

[1] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, July 1989.

[2] N. D. Zervas and et al., "Evaluation of design alternatives for the 2-d-discrete wavelet transform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1246–1262, Dec. 2001.

[3] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Ap-*

plied and Computaional Harmonic Analysis, vol. 3, no. 15, pp. 186–200, 1996.

[4] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *The Journal of Fourier Analysis and Applications*, vol. 4, pp. 247–269, 1998.

[5] C. Chrysafis and A. Ortega, "Line-based, reduced, wavelet image compression," *IEEE Transactions on Image processing*, vol. 9, no. 3, pp. 378–389, Mar. 2000.

[6] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 1, no. 2, pp. 191–202, June 1993.

[7] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: From single chip implementations to mappings on SIMD array computers," *IEEE Transactions on Signal Processing*, vol. 43, no. 3, pp. 759–771, Mar. 1995.

[8] J. Reichel, "On the arithmetic and bandwidth complexity of the lifting scheme," in *Proc. of International Conference on Image Processing*, 2001, pp. 198–201.

[9] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Efficient VLSI architectures of lifting-based discrete wavelet transform by systematic design method," in *IEEE International Symposium on Circuits and Systems, to appear*, 2002.

[10] P.-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.

[11] David B. H. Tay, "A class of lifting based integer wavelet transform," in *Proc. of International Conference on Image Processing*, 2001, vol. 1, pp. 602–605.

[12] Z. Guangjun, C. Lizhi, and C. Huowang, "A simple 9/7-tap wavelet filter based on lifting scheme," in *Proc. of International Conference on Image Processing*, 2001, vol. 2, pp. 249–252.