AN EFFICIENT ARCHITECTURE FOR TWO-DIMENSIONAL INVERSE DISCRETE WAVELET TRANSFORM

Po-Cheng Wu [1], Chao-Tsung Huang [2], and Liang-Gee Chen [2]

[1] Network Communication Lab., Institute for Information Industry, Taipei, Taiwan, R. O. C.
[2] Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, R. O. C.
E-mail: [1] pcwu@netrd.iii.org.tw, [2] {cthuang, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

This paper proposes an efficient architecture for the two-dimensional inverse discrete wavelet transform (2-D IDWT). The proposed architecture includes an inverse transform module, a RAM module, and a multiplexer. In the inverse transform module, we employ the coefficient folding technique and the polyphase decomposition technique to the interpolation filters of stages 1 and 2, respectively. The RAM size is N/2 × N/2. The advantages of the proposed architecture are the 100% hardware utilization, fast computing time, regular data flow, and low control complexity, making this architecture suitable for next generation image coding/decoding systems.

1. INTRODUCTION

With the rapid progress of VLSI design technologies, many processors based on audio and image signal processing have been developed recently. The Two-Dimensional Discrete Wavelet Transform (2-D DWT) plays a major role in the JPEG-2000 image compression standard [1]. The architecture of the 2-D DWT is mainly composed of the multirate filters. Because extensive computation is involved in the practical applications, e.g., digital cameras, high efficiency and low cost hardware is indispensable.

At present, many VLSI architectures for the 2-D DWT have been proposed [2]-[7] to meet the requirements of real time processing. However, there has been very little research on the topic of designing efficient architectures for the 2-D IDWT. Therefore, in this paper, we propose an efficient 2-D IDWT architecture for VLSI implementation. The advantages of the proposed architecture are the 100% hardware utilization, fast computing time, regular data flow, and low control complexity. Additionally, because of the regular structure, the proposed architecture can easily be scaled with the filter length and the 2-D IDWT level.

This paper is organized as follows. Section 2 proposes an efficient architecture for the 2-D IDWT. Section 3 lists the performance of the proposed 2-D IDWT architecture. Finally, we give discussions in Section 4.

2. PROPOSED 2-D IDWT ARCHITECTURE

The block diagram of the proposed 2-D IDWT architecture is shown in Fig. 1, which includes an inverse transform module, a RAM module, and a multiplexer. The size of the RAM module is $N/2 \times N/2$. The reconstruction scheme is level by level and described as follows. In the first level reconstruction, the multiplexer selects the $(LL)^{J-1}LL$ band from the input. The inverse transform module reconstructs the $(LL)^{J-2}LL$ band from $(LL)^{J-1}LL$, $(LL)^{J-1}LH$, $(LL)^{J-1}HL$, and $(LL)^{J-1}HH$ bands, and saves the $(LL)^{J-2}LL$ band to the RAM module. In the second level reconstruction, the multiplexer selects the $(LL)^{J-2}LL$ band from the RAM module. The inverse transform module reconstructs the $(LL)^{J-3}LL$ band from $(LL)^{J-2}LL$, $(LL)^{J-2}LH$, $(LL)^{J-2}HL$, and $(LL)^{J-2}HH$ bands, and saves the $(LL)^{J-3}LL$ band to the RAM module. In the third level reconstruction, the multiplexer selects the $(LL)^{J-3}LL$ band from the RAM module. The inverse transform module reconstructs the $(LL)^{J-4}LL$ band from $(LL)^{J-3}LL$, $(LL)^{J-3}LH$, $(LL)^{J-3}HL$, and $(LL)^{J-3}HH$ bands, and saves the $(LL)^{J-4}LL$ band to the RAM module. This procedure repeats until the Jth level (i.e., the last level) reconstruction. In the last level reconstruction, the multiplexer selects the LL band from the RAM module. The inverse transform module reconstructs the original image from LL, LH, HL, and HH bands, and outputs the reconstruction image. The sizes of the $(LL)^{J-1}LL$, $(LL)^{J-2}LL$, $(LL)^{J-3}LL$, ..., $LLLL$, LL bands, and the reconstruction image are $N/2^J \times N/2^J$, $N/2^{J-1} \times N/2^{J-1}$, $N/2^{J-2} \times N/2^{J-2}$, ..., $N/4 \times N/4$, $N/2 \times N/2$, and $N \times N$, respectively.

The advantage of such a scheme is that the data flow is very regular. We can concentrate our effort to efficiently design the inverse transform module. As shown in Fig. 2, the inverse transform module is tree-structured and comprises two stages. Stage 1 performs horizontal

0-7803-7448-7/02/$17.00 ©2002 IEEE

filtering, and stage 2 performs vertical filtering. To design the inverse transform module efficiently, we assume "***a***" to be the area cost and "***t***" to be the time cost required in stage 2. According to the original design as shown in Fig. 2, the number of filters required in stage 1 is double that of stage 2. That is, $2a$ is the area cost required in stage 1. On the other hand, because of the interpolation operation before the filters of stage 2, the quantity of data for filtering in each filter of stage 2 is double that of stage 1. Hence, the computing time required in stage 1 is half of t, i.e., $t/2$. Because stage 2 is cascaded after stage 1, stage 1 can not generate outputs until stage 2 finishes its job. Therefore, we find that there will be $2a \times (t - t/2) = at$ hardware idle in stage 1. In other words, the hardware utilization in the original design is inefficient.

In order to solve this problem, we consider a single interpolation filter. The interpolation filter can be implemented directly by a two-folded interpolator followed by a filter. However, the interpolator inserts a zero between each adjacent pair of samples at the filter input, causing poor hardware utilization. Hence, we employ two different design techniques, which can achieve the full hardware utilization to enhance its performance. The first technique is the polyphase decomposition technique as illustrated in Fig. 3, which decomposes the filter coefficients into even-ordered and odd-ordered parts. The input data are multiplied with the even-ordered and odd-ordered coefficients simultaneously. The output data are obtained from the even part in the even clock cycles and from the odd part in the odd clock cycles. The internal clock rate is reduced to half after employing the polyphase decomposition technique. Therefore, we can double the input clock rate to increase the throughput. When the quantity of processing data is the same, the computation time will be reduced to half. Thus, this technique can reduce the time cost from T to $T/2$. We use the symbol $T/2$ to represent the polyphase decomposition technique. Table 1 shows the data flow of the interpolation filter employing the polyphase decomposition technique.

The second technique is the coefficient folding technique. As illustrated in Fig. 4, every two coefficients share one set of a multiplier, adder, and register. The switches select the filter coefficients. The input data are multiplied with the even-ordered coefficients in the even clock cycles and with the odd-ordered coefficients in the odd clock cycles. Because every two coefficients share one set of a multiplier, adder, and register, this technique can approximately reduce the area cost from A to $A/2$. We use the symbol $A/2$ to represent the coefficient folding technique. Table 2 shows the data flow of the interpolation filter employing the coefficient folding technique.

Now, we employ these two design techniques to the interpolation filters of stages 1 and 2, respectively. Hence, four different design methods are derived for the inverse transform module. The design strategy (including the original design) is listed in Table 3. From Table 3, we find that if we employ the coefficient folding technique to stage 1 and the polyphase decomposition technique to stage 2, the area and time cost will both be the same a and $t/2$ in stages 1 and 2. Thus, the total area cost is $2a$ and the total time cost is $t/2$. The AT product is reduced from $3at$ to at, and no hardware is idle in stage 1. Therefore, the performance of the new design method is three times more efficient than the original design. In contrast, the other design methods, as listed in Table 3, cause the hardware to be idle in stage 1. Hence, they are not efficient design schemes.

Assume that the low-pass filter has four taps: a_0, a_1, a_2, and a_3, and the high-pass filter has four taps: b_0, b_1, b_2, and b_3. Fig. 5 illustrates the inverse transform module employing both the coefficient folding and the polyphase decomposition techniques. In stage 2 of the inverse transform module, because the image data are fed by a raster scan mode, each coefficient requires a line delay to store the row data for vertical filtering. The size of the line delay is N. Thus, in the Jth level (i.e., the last level) reconstruction, it can store the row data output from the interpolation filters of stage 1. In the different reconstruction levels, the select signals change the size of the line delay to $N/2^{J-1}$ in the first level, $N/2^{J-2}$ in the second level, $N/2^{J-3}$ in the third level, $N/2^{J-4}$ in the fourth level, ... , $N/4$ in the (J-2)th level, $N/2$ in the (J-1)th level, and N in the Jth level.

We have assumed that the filters in stages 1 and 2 have the same length. However, this condition is not necessary for the correct operation in practice. In addition, Because of the regular structure, the proposed architecture can be easily scaled with the filter length and the 2-D IDWT level.

3. PERFORMANCE OF THE PROPOSED ARCHITECTURE

Table 4 lists the performance of the proposed 2-D IDWT architecture in terms of the number of multipliers, the number of adders, storage size, control complexity, and hardware utilization. Concerning the storage size, the proposed architecture requires a RAM module of size $N/2 \times N/2$ to save the intermediate data. However, because the proposed architecture is mainly applied in the image coding/decoding systems, it can use the memory already existing in the systems to save the intermediate data. Hence, in this condition, the proposed architecture will not require the RAM module and the value $N^2/4$ can be discarded in Table 4.

4. CONCLUSIONS

In recent years, many VLSI architectures for the 2-D DWT have been proposed to meet the requirements of real time processing. However, there has been very little research on the topic of designing efficient architectures for the 2-D IDWT. Therefore, in this paper, we have proposed an efficient 2-D IDWT architecture. The proposed architecture has been correctly verified by the Verilog Hardware Description Language (Verilog HDL). The advantages of the proposed architecture are the 100% hardware utilization, fast computing time, regular data flow, and low control complexity, making this design suitable for next generation image coding/decoding systems, e.g., JPEG-2000.

5. REFERENCES

[1] *Coding of Still Pictures: JPEG 2000 Verification Model Version 7.0*, ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG8), Apr. 2000.

[2] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. VLSI Systems*, vol. 1, no. 2, pp. 191-202, June 1993.

[3] M. Vishwanath, R. Owens, and M. J. Irwin, "VLSI architecture for the discrete wavelet transforms," *IEEE Trans. Circuits and Systems II*, vol. 42, no. 5, pp. 305-316, May. 1995.

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

[5] J. C. Limqueco and M. A. Bayoumi, "A VLSI architecture for separable 2-D discrete wavelet transforms," *Journal of VLSI Signal Processing*, vol. 18, pp. 125-140, 1998.

[6] G. Lafruit, F. Catthoor, J. P. H. Cornelis, and H. J. D. Man, "An efficient VLSI architecture for 2-D wavelet image coding with novel image scan," *IEEE Trans. VLSI Systems*, vol. 7, no. 1, pp. 56-68, Jan. 1999.

[7] P. C. Wu and L. G. Chen, "An efficient architecture for two-dimensional discrete wavelet transform," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 536-545, Apr. 2001.

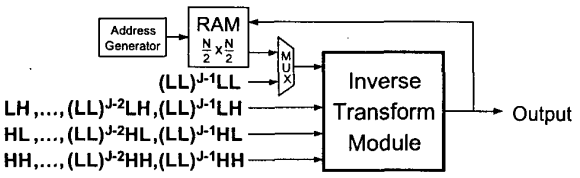

Fig. 1. The proposed 2-D IDWT architecture.

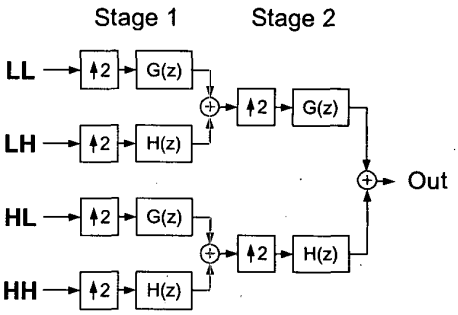

Fig. 2. The tree-structured inverse transform module.

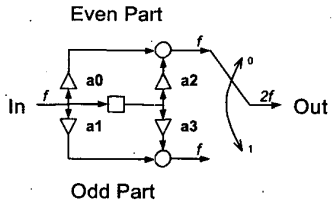

Fig. 3. The interpolation filter employing the polyphase decomposition technique.

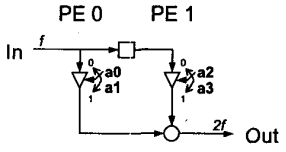

Fig. 4. The interpolation filter employing the coefficient folding technique.

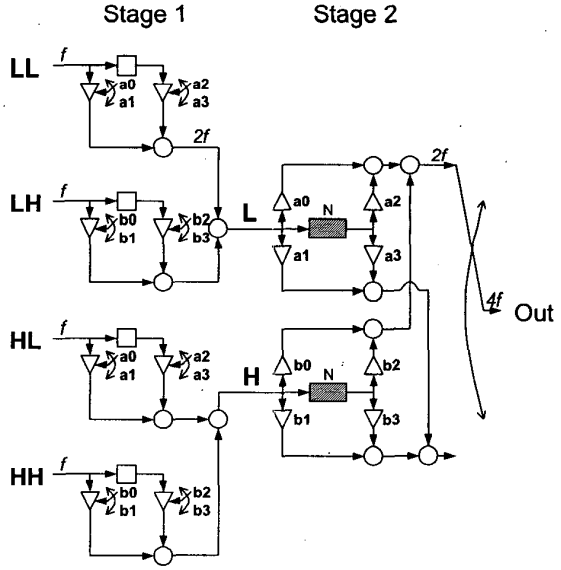

Fig. 5. The inverse transform module employing the coefficient folding technique to stage 1 and the polyphase decomposition technique to stage 2.

Clk	SW	In	Even Part	Odd Part	Out
0	0	$x(0)$	$a_0x(0)$	$a_1x(0)$	$a_0x(0)$
1	1				$a_1x(0)$
2	0	$x(1)$	$a_0x(1)+a_2x(0)$	$a_1x(1)+a_3x(0)$	$a_0x(1)+a_2x(0)$
3	1				$a_1x(1)+a_3x(0)$
4	0	$x(2)$	$a_0x(2)+a_2x(1)$	$a_1x(2)+a_3x(1)$	$a_0x(2)+a_2x(1)$
5	1				$a_1x(2)+a_3x(1)$
6	0	$x(3)$	$a_0x(3)+a_2x(2)$	$a_1x(3)+a_3x(2)$	$a_0x(3)+a_2x(2)$
7	1				$a_1x(3)+a_3x(2)$
8	0	$x(4)$	$a_0x(4)+a_2x(3)$	$a_1x(4)+a_3x(3)$	$a_0x(4)+a_2x(3)$
9	1				$a_1x(4)+a_3x(3)$

Table 1. The data flow of the interpolation filter employing the polyphase decomposition technique. (SW: the directions of the switch.)

Clk	SW	In	PE 0	PE 1	Out
0	0	$x(0)$	$a_0x(0)$		$a_0x(0)$
1	1		$a_1x(0)$		$a_1x(0)$
2	0	$x(1)$	$a_0x(1)$	$a_2x(0)$	$a_0x(1)+a_2x(0)$
3	1		$a_1x(1)$	$a_3x(0)$	$a_1x(1)+a_3x(0)$
4	0	$x(2)$	$a_0x(2)$	$a_2x(1)$	$a_0x(2)+a_2x(1)$
5	1		$a_1x(2)$	$a_3x(1)$	$a_1x(2)+a_3x(1)$
6	0	$x(3)$	$a_0x(3)$	$a_2x(2)$	$a_0x(3)+a_2x(2)$
7	1		$a_1x(3)$	$a_3x(2)$	$a_1x(3)+a_3x(2)$
8	0	$x(4)$	$a_0x(4)$	$a_2x(3)$	$a_0x(4)+a_2x(3)$
9	1		$a_1x(4)$	$a_3x(3)$	$a_1x(4)+a_3x(3)$

Table 2. The data flow of the interpolation filter employing the coefficient folding technique. (SW: the directions of the switches.)

Methods		Stage 1		Stage 2		Total	Total	AT	Stage 1
Stage 1	Stage 2	Area	Time	Area	Time	Area	Time	Prod.	Idle
Original Design		$2a$	$t/2$	a	t	$3a$	t	$3at$	at
$T/2$	$T/2$	$2a$	$t/4$	a	$t/2$	$3a$	$t/2$	$3at/2$	$at/2$
$A/2$	$A/2$	a	$t/2$	$a/2$	t	$3a/2$	t	$3at/2$	$at/2$
$T/2$	$A/2$	$2a$	$t/4$	$a/2$	t	$5a/2$	t	$5at/2$	$3at/2$
$A/2$	$T/2$	a	$t/2$	a	$t/2$	$2a$	$t/2$	at	**0**

Table 3. The design strategy of the inverse transform module. ($T/2$: the polyphase decomposition technique, and $A/2$: the coefficient folding technique.)

Architecture	Multipliers	Adders	Storage Size	Computing Time	Control Complexity	Hardware Utilization
Ours	$4K$	$4K$	$N^2/4+(K-2)N+2K-4$	$0.5N^2$~$0.67N^2$	Simple	100%

Table 4. The performance of the proposed 2-D IDWT architecture. (K: the filter length, and N^2: the image size.)