EFFICIENT VLSI ARCHITECTURES OF LIFTING-BASED DISCRETE WAVELET TRANSFORM BY SYSTEMATIC DESIGN METHOD

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.
E-mail: {cthuang, pctseng, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

In this paper, an effective systematic design method is proposed to construct several efficient VLSI architectures of 1-D and 2-D lifting-based discrete wavelet transform. This design method first performs a specific lifting factorization for any finite discrete wavelet transform filter to obtain an optimal algorithm representation for hardware implementation. The optimized algorithm then turns into 1-D systolic architectures through dependence graph formation and systolic arrays mapping. Based on the 1-D architectures, a general 2-D discrete wavelet transform framework is used to construct the corresponding 2-D architectures. According to the comparison results, the constructed VLSI architectures are more efficient than previous arts in term of arithmetic units and memory storage.

1. INTRODUCTION

During the past decade, wavelets have been developed as an effective multiresolution analysis tool. Since the discrete wavelet transform (DWT) was deduced by Mallat [1], many researches on wavelet-based image analysis and compression have derived fruitful results. The DWT can decompose a signal into different subbands of well-defined time-frequency characteristics. For instance, in the dyadic type decomposition, the lower frequency subbands have finer frequency resolution and coarser time resolution. Emerging image coding standards, such as the JPEG2000 still image coding and the MPEG-4 still texture coding, have adopted the DWT as their transform coder, due to its well time-frequency decomposition. However, the DWT requires much more arithmetic computations than old-fashioned transforms such as the discrete cosine transform (DCT). Besides, contrary to the block-based DCT, the DWT is basically frame-based. This huge amount of memory requirement is usually the bottleneck of the implementation for 2-D DWT.

Thanks to the appearance of lifting scheme [2] and a factorization method that factors wavelet transforms into lifting steps [3], the lifting scheme is widely used to speed up the DWT computation and possible to reduce the memory requirement of 2-D DWT. However, due to the exploitation of Euclidean algorithm for Laurent polynomials in this lifting factorization method, the factorization process is non-unique. This is, there may exist many essentially different lifting factorizations, but which one more suitable for software or hardware implementation is still an open design issue.

In this paper, an effective systematic design method is proposed to construct several efficient VLSI architectures of 1-D and 2-D lifting-based DWT. The concepts of lifting scheme and the lifting factorization method are described in section 2. Section 3 presents the detailed design stages of proposed systematic design method. A case study is given in section 4 for an example. In section 5, the constructed architectures are compared with several previous arts to show its efficiency. Finally, brief summaries are given in section 6 to conclude this paper.

2. DISCRETE WAVELET TRANSFORM WITH LIFTING SCHEME

The lifting scheme is a new method for constructing wavelets entirely by spatial approach [2]. Using lifting scheme to construct wavelets has many advantages, such as allowing a faster and fully in-place implementation of the wavelet transforms, immediately to find the inverse transform, easily to manage the boundary extension, and possibly of defining a wavelet-like transform that maps integer-to-integer. According to [3], any DWT with finite filter can be decomposed into a finite sequence of simple filtering steps, which is called the 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. Fig. 1 shows the general block diagram of a DWT filter. The forward transform uses two analysis filters $\tilde{h}$ (low pass) and $\tilde{g}$ (high pass) followed by subsampling, while the inverse transform first upsamples and then uses two synthesis filters h (low pass) and g (high pass).

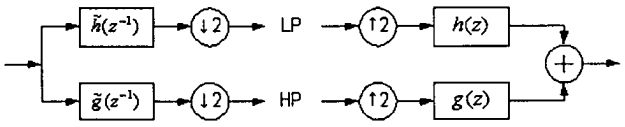


Figure 1: General block diagram of a DWT filter

Since the polyphase representation of a filter h is

$$h(z)=h_e(z^2)+z^{-1}h_o(z^2)$$

the polyphase matrix of a DWT filter can be assembled as

$$P(z)=\begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix}$$

In [3], it has been shown that if h and g is a complementary filter pair, then with the exploitation of Euclidean algorithm for Laurent polynomials, there always exist Laurent polynomials $s_i(z)$, $t_i(z)$ and a non-zero constant K so that

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

In other words, any finite DWT filter can be obtained by starting with the Lazy wavelet followed by several lifting steps with a scaling.

3. PROPOSED SYSTEMATIC DESIGN METHOD

In this section, a systematic design method for hardware implementation of lifting-based DWT is presented. By this systematic design method, several efficient VLSI architectures of 1-D and 2-D lifting-based DWT can be easily constructed. As shown in Fig. 2, this design method consists of several design stages. Once a finite DWT filter is chosen, four subsequent design stages are then performed to construct the corresponding VLSI architectures. Detailed contents of each design stage are described in following subsections.

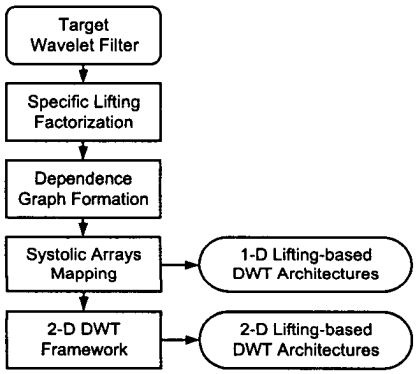


Figure 2: Proposed systematic design method

3.1. Specific Lifting Factorization

As pointed out in section 1, the lifting factorization process is non-unique. This freedom diversifies the design space of hardware implementation for lifting-based DWT. In the proposed systematic design method, a specific lifting factorization is chosen for all target wavelet filters. This factorization principle is to factor the Laurent polynomials $s_i(z)$ and $t_i(z)$ as symmetric or anti-symmetric as possible and allow at most two coefficients in each lifting step. Moreover, one lifting step can further be decomposed into two lifting steps as

$$\begin{bmatrix} 1 & a(z)+b(z) \\ 0 & 1 \end{bmatrix}=\begin{bmatrix} 1 & a(z) \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & b(z) \\ 0 & 1 \end{bmatrix}$$

Following this principle, in each lifting step, an even location will only get information from two odd locations or vice versa. There exist only four possible categories of basic processing element in such factorization as shown in Fig. 3. Fig. 4 shows the four possible lifting step categories for $t_i(z)$, and each corresponds to the four basic processing element categories in Fig. 3. The case of $s_i(z)$ is similar.

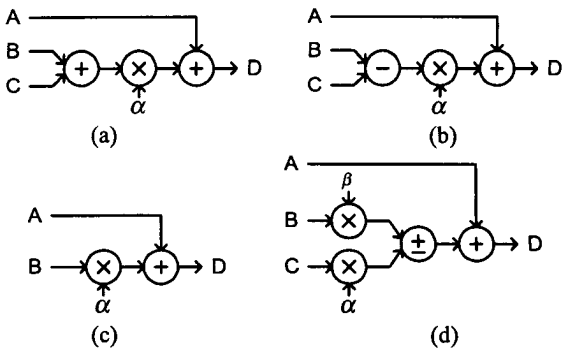


Figure 3: Four categories of basic processing element

$$\underset{\text{(a)}}{\begin{bmatrix} 1 & 0 \\ \alpha(1+z^{-1}) & 1 \end{bmatrix}} \quad \underset{\text{(b)}}{\begin{bmatrix} 1 & 0 \\ \alpha(1-z^{-1}) & 1 \end{bmatrix}} \quad \underset{\text{(c)}}{\begin{bmatrix} 1 & 0 \\ \alpha & 1 \end{bmatrix}} \quad \underset{\text{(d)}}{\begin{bmatrix} 1 & 0 \\ \alpha \pm \beta z^{-1} & 1 \end{bmatrix}}$$

Figure 4: Four lifting step categories for $t_i(z)$

The category (d) in Fig. 3 and Fig. 4 can be regarded as a general case of (a), (b), and (c). One can expect that, if the target wavelet filter is linear phase, namely, symmetric or anti-symmetric, then only the first three categories should appear by specific lifting factorization. In such cases, the number of multiplication in each lifting step can be reduced by at most a factor of two. In the following design stages, the scale factor K and 1/K will not be involved since it can be implemented exactly with two multipliers.

3.2. Dependence Graph Formation

Once the specific lifting factorization is done, a dependence graph (DG) can be drawn for its corresponding lifting factored wavelet filter. However, in order to simplify the complexity of next design stage, the Systolic Arrays Mapping, a specific formation of the DG is performed to obtain a more regular and compact DG form.

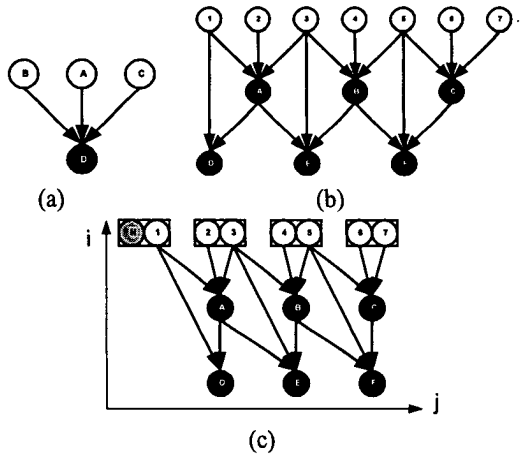


Figure 5: Dependence graph formation

As shown in Fig. 5(a), any lifting step constructed by specific lifting factorization can be depicted as a general basic DG that is a combination of three input nodes (A, B, C) and one computation-output node (D). Due to the step-by-step property of specific lifting factorization, without loss of generality but for simplicity and regularity consideration, one slice of a DG can be depicted as shown in Fig. 5(b). In Fig. 5(b), the white node (tagged 1~7) denoted as the input node, and the black node (tagged A~F) denoted as the computation-output node. The formation principle is described as following two steps: First, merge one pair of even and odd input nodes into a new input node. As shown in Fig. 5(c), except for the first even node (tagged 1), one even and one odd node are merged into new single input node. The first even node 1 can be treated as merged with a virtual odd node N such that this merging step is regular. This step can make sure that the following systolic arrays mapped architectures have unified input-output ports and throughput. Second, move the computation-output nodes to suitable position such that there is no backward directional data flow existing in the DG. This step can make sure that the mapped architectures have unified data flow direction.

3.3. Systolic Arrays Mapping

After the DG formation design stage, an unique systolic arrays mapping parameters are applied to the DG to obtain the corresponding signal flow graph (SFG). As the same systolic architecture definitions in [4], the DG in Fig. 5(c) is mapped by

$$\textit{Processor Vector } p = (0,1)^T$$

$$\textit{Projection Vector } d = (1,0)^T$$

$$\textit{Scheduling Vector } s = (1,0)^T$$

The resulting SFG is depicted as shown in Fig. 6(a). The detailed architecture of each PE can be found in Fig. 3, one of the four categories of basic processing element dependent on its corresponding lifting step category. In this architecture, the critical path is two PE delay. If better performance is required, then further pipelining can be applied to the original SFG. As shown in Fig. 6(b), after the pipelining (dash line), two pipeline delay registers (D) are added but one PE critical path delay is achieved. By above three design stages, 1-D lifting-based architectures of any finite DWT filter can be easily constructed. Fig. 7 shows the general model of 1-D lifting-based DWT architectures constructed by proposed systematic design method.

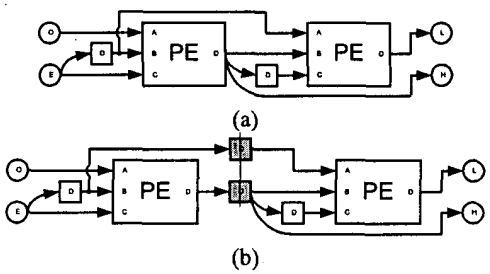


Figure 6: Systolic arrays mapped 1-D DWT architectures

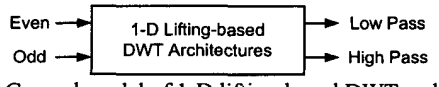


Figure 7: General model of 1-D lifting-based DWT architectures

3.4. 2-D DWT Framework

The last design stage of proposed systematic design method is to put the constructed 1-D lifting-based DWT architectures into a general 2-D DWT framework to construct the 2-D lifting-based DWT architectures.

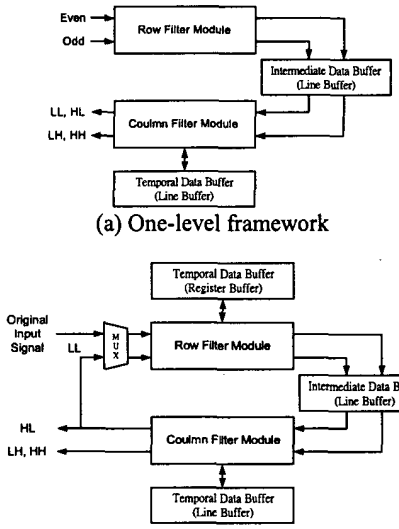


Figure 8: The general 2-D DWT framework

The proposed general 2-D framework for lifting-based DWT shown in Fig. 8 uses the line-based architecture. Fig. 8(a) is the one-level framework, and the row and column filter modules in this figure are previously discussed 1-D lifting-based DWT architectures. The intermediate data buffer is composed of six separate two-port memories, three for row filtered low pass signal and three for row filtered high pass signal. Each memory depth is half the signal line width. These intermediate data memories act as dynamic rotation buffers. At each time moment, two intermediate data memories of low pass or high pass signal are accessed by the column filter module, and one pair of intermediate data memories are accessed by the row filter module. The temporal data buffer is composed of several separate two-port memories, each depth is the same as the signal line width. The number of temporal data memories is determined by the amount of temporal registers in 1-D lifting-based DWT architectures. For the two cases in Fig. 6, (a) requires only two and (b) requires two more for pipeline registers

The one-level framework can be easily extended to multi-level framework in dyadic decomposition type scheduled by recursive pyramid algorithm [5] (RPA) as shown in Fig. 8(b). Instead of two-input per cycle in the case of one-level framework, the multi-level framework only requires one-input per cycle by arranging the RPA schedule for row and column filter module so as to make this framework more feasible. Assume the decomposition level is J, then due to the multi-level operation, the intermediate and temporal data buffers are also determined by the J. Besides, for the row filter module, J-1 times of the amount of temporal registers in 1-D lifting-based DWT architectures are required as its temporal data buffer. Hence, the total amount of memory storage can be summarized as

$$Storage = (3+T_C)\times N\times(1+\frac{1}{2}+...+\frac{1}{2^{J-1}})+J\times T_R$$

where N denotes the signal line width as well as T_R and T_c denote the amount of temporal registers in 1-D row and column lifting-based DWT architectures, respectively. The line buffer for intermediate data and column temporal data can be easily implemented with feasible two-port memories because of very regular memory addressing.

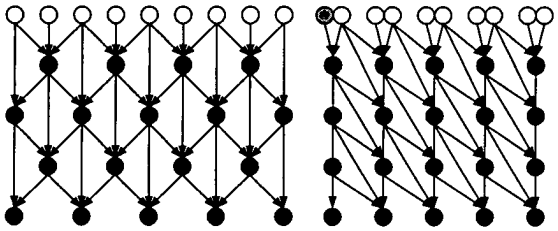
4. CASE STUDY

In this section, a practical example is given to show the effective of proposed systematic design method. The case to be studied is the popular (9,7) odd symmetric biorthogonal filter, which is adopted by JPEG2000 lossy coding. First, by specific lifting factorization, the polyphase matrix can be factored into four lifting steps and a scaling constant.

$$P(z)=\begin{bmatrix}1 & \alpha(1+z^{-1})\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \beta(1+z) & 1\end{bmatrix}\begin{bmatrix}1 & \gamma(1+z^{-1})\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \delta(1+z) & 1\end{bmatrix}\begin{bmatrix}\zeta & 0\\ 0 & \frac{1}{\zeta}\end{bmatrix}$$

α = -1.586134342; β = -0.05298011854; γ = 0.8829110762;
δ = 0.4435068522; ζ = 1.149604398

This factorization leads to the original and specific DG formation as shown in Fig. 9(a) and (b).



(a) (b)

Figure 9: Dependence graph formation for (9,7) filter

After the systolic arrays mapping, the 1-D lifting-based DWT architecture for (9,7) filter is shown in Fig. 10. The PE architecture in this figure corresponds to Fig. 3(a).

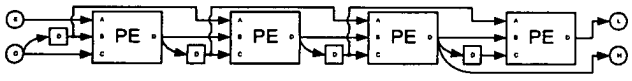


Figure 10: 1-D lifting-based DWT architecture for (9,7) filter

In above architecture, the number of temporal registers is four. Of course, if better performance is required, further pipelining can be made among each PE. However, since the number of temporal registers in 1-D lifting-based DWT architectures determines the total storage of 2-D DWT architecture, some trade-off should be considered to achieve an optimal solution for target applications.

5. COMPARISONS

In order to show the efficiency of constructed lifting-based DWT architectures, several 1-D and 2-D DWT architectures are chosen for comparison. Here (9,7) filter is used as the target DWT filter. In [6], a 1-D lifting-based DWT architecture for (9,7) is proposed. Focus only on the lifting-based DWT datapath and ignoring the scale factors, the comparison results are shown in Table 1. Although the improvement is not significant, but note that our design method can retarget for any finite DWT filter.

Table 1: Comparison with 1-D lifting-based DWT architecture

Architecture (1-D)	Multiplier	Adder	Register
Lifting [6]	4	8	6
Proposed	4	8	4

A 2-D architecture comparison with the well-known systolic-parallel [7] and parallel-parallel architectures [8] is given in Table 2.

Table 2: Comparison with 2-D DWT architectures

Architecture (2-D)	Multiplier	Adder	Line Buffer
Systolic-Parallel [7]	36	28	19N
Parallel-Parallel [8]	36	28	19N
Proposed	10	16	14N

6. CONCLUSION

We have presented an effective and systematic design method to construct the VLSI architectures of lifting-based DWT. This complete design flow can obtain feasible and efficient architectures for any finite DWT filter by mapping lifting steps to systolic arrays. Besides, the proposed lifting strategy can achieve optimal decomposition for linear phase filter banks. Furthermore, since the lifting factorization algorithm can decompose any 2-channel filter bank of perfect reconstruction very well, our design method and proposed architectures are also suitable for any 2-channel subband filter bank.

7. REFERENCE

[1] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 11, pp. 674-693, July 1989.

[2] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Applied and Computational Harmonic Analysis 3*, pp. 186-200, 1996.

[3] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *The J. of Fourier Analysis and Applications*, Vol. 4, pp. 247-269, 1998.

[4] K. K. Parhi, VLSI Digital Signal Processing Systems – Design and Implementation, Chapter 7: Systolic Architecture Design

[5] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," *IEEE Transactions on Signal Processing*, Vol. 42, No. 3, pp. 673-677, March 1994.

[6] J. M. Jou, Y. H. Shiau, C. C. Liu, "Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme" *IEEE International Symposium on Circuits and Systems 2001*, Vol. 2, pp. 529 -532

[7] M. Vishwanath, R. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," *IEEE Trans. Circuits and Systems II, Analog and Digital Signal Processing*, Vol. 42, No. 5, pp. 305-316, May 1995.

[8] 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 Trans. Signal Processing*, Vol. 43, No. 3, pp. 759-771, March 1995.