VLSI implementation of shape-adaptive discrete wavelet transform

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

DSP/IC Design Lab, Graduate Institute of Electronics Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

ABSTRACT

In this paper, several VLSI architectures and implementations of Shape-Adaptive Discrete Wavelet Transform (SA-DWT) with odd symmetric biorthogonal filters are presented. The hardware implementation issues of SA-DWT algorithm are first addressed, and one lifting scheme together with some appropriate shape information processing units is introduced for the 1-D SA-DWT architectures. These architectures can efficiently perform 1-D length-adaptive DWT on continuous line segments and solve the corresponding boundary extension and subsampling problems without any overhead. In addition to 1-D architectures, two 2-D SA-DWT architectures are also proposed. One is the direct method, which uses the 1-D SA-DWT architectures as basis to extend to 2-D SA-DWT by direct row-column approach. The other is the line-based method, which is an efficient and feasible architecture that uses internal line buffers rather than external frame memory. These two 2-D SA-DWT architectures are both VLSI implemented by using cell-based chip design flow. The Daubechies (9,7) filter is used for the implementation by direct method, and the Daubechies (9,3) filter is used for the line-based implementation. The proposed architectures can be applied both in arbitrarily shaped visual objects coding such as MPEG-4 still texture coding and rectangular region image coding such as JPEG2000 still image coding.

Keywords: shape-adaptive discrete wavelet transform, VLSI implementation.

1. INTRODUCTION

Discrete Wavelet Transform (DWT) is an important tool for signal processing and analysis due to its well time-frequency decomposition [1]. It has been widely applied in visual coding applications, and several emerging standards such as JPEG2000 still image coding [2] and MPEG-4 still texture coding [3] have adopted DWT as their transform coder. The ability to compress visual objects is a very important feature in the next generation visual coding standards such as MPEG-4, since it provides flexible functionality to manipulate visual objects in multimedia applications and could improve coding efficiency in very low bit-rate coding. There have been many research efforts in developing new algorithms for efficiently coding of arbitrarily shaped visual objects. Among them, the Shape-Adaptive Discrete Wavelet Transform (SA-DWT) proposed by [4] could be the best one and has been adopted in MPEG-4 standard due to its wonderful efficiency. The number of coefficients after SA-DWT is identical to the number of pixels in the original arbitrarily shaped visual object. Besides, the spatial correlation, locality properties of wavelet transforms, and self-similarity across subbands are well preserved in SA-DWT. For a rectangular region, the SA-DWT becomes identical to the conventional wavelet transforms. These features make the SA-DWT achieve high coding efficiency while satisfying the functionality of representing both arbitrarily shaped and rectangular region visual texture.

Although there have been many efforts in developing the architectures of the DWT, few mentioned how to perform the boundary extension of a very short signal, which is an important issue in the SA-DWT. Besides, most of the implementations of SA-DWT in the literature are based on software approach such as [5]. If real-time requirement is necessary, then dedicated hardware solution must be considered. The rest of this paper is organized as follows. In the next section, several important algorithms including SA-DWT and lifting scheme are first discussed. The proposed architectures of 1-D SA-DWT and 2-D SA-DWT are then proposed in section 3 and 4, respectively. In section 5, the VLSI implementations of two proposed 2-D SA-DWT architectures are presented. Finally, a brief summary is given to conclude this paper.

Visual Communications and Image Processing 2002,
C.-C. Jay Kuo, Editor, Proceedings of SPIE Vol. 4671 (2002) © 2002 SPIE · 0277-786X/02/$15.00

2. ALGORITHMS

2.1. Shape-Adaptive Discrete Wavelet Transform

The main idea of SA-DWT algorithm is to treat the arbitrarily shaped visual object as several continuous line segments. 1-D row-wise length-adaptive DWT is first performed on each line segment according to the corresponding shape information and then followed by 1-D column-wise length-adaptive DWT. Fig. 1 shows the conditions of 1-D row-wise length-adaptive DWT performed on three continuous line segments of a visual object (the three white lines).

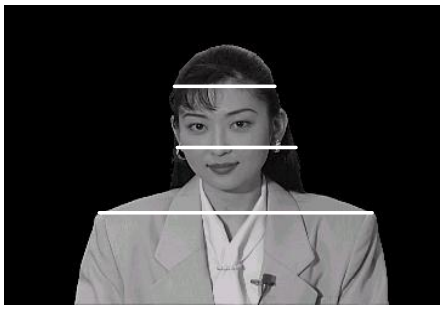


Figure 1: 1-D row-wise length-adaptive DWT

As discussed as in [4], for different wavelet filters such as orthogonal, odd symmetric biorthogonal, and even symmetric biorthogonal filters, different boundary extension types are used for processed line segments. In this paper, we consider the odd symmetric biorthogonal filters only for its best coding efficiency, but other wavelet filters can also be extended in the same way. Therefore, as shown in Fig. 2, there are two possible types of boundary extension under consideration. For any odd symmetric biorthogonal wavelet filters, boundary extension type B is used at the leading and trailing boundaries in forward transform and at the leading boundaries in inverse transform. However, which type of boundary extension should be used at the trailing boundaries in inverse transform will depend on if the signal length is odd or even.

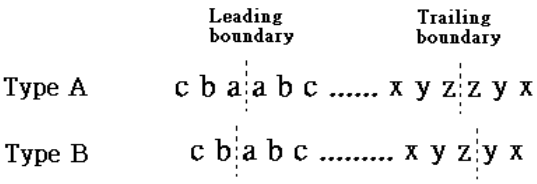


Figure 2: Boundary extension types

Also as discussed as in [4], There are two subsampling strategies in SA-DWT: one is global subsampling and the other is local subsampling. For very low bit-rate coding, local subsampling strategy could achieve better coding efficiency, but global subsampling strategy achieves higher signal processing gain. In this paper, we adopt global low pass even subsampling and high pass odd subsampling strategy. That is, which kind of boundary extension and subsampling strategy should be performed for each processed line segment will depend on the position of the segment in the global frame. As the boundary extension condition, other subsampling strategies can also be extended in the same way.

There is one special case that we want to emphasize. If the line segment has only one point, then one lowpass coefficient should be produced no matter even or odd position it locates in the global frame. However, in inverse SA-DWT, is would be difficult to identify the one single low pass coefficient that locates in an even or odd position. The solution to solve this problem will affect the following backward entropy coding process. In this paper, we add one another shape mask information for SA-DWT coefficients. This one-bit information can be used to pass the corresponding location of one point line segment in inverse SA-DWT. Moreover, this additional bit plane can be encoded very efficiently if the special case of one point line segment does not happen very often.

After the discussion of the concepts and algorithms of SA-DWT, we can summary several hardware implementation issues of SA-DWT. First, the filtering operations of DWT computation must be performed as required. Second, complex boundary extension conditions belonging to each line segment must be solved without overhead. Third, suitable operations for target subsampling strategy should also be considered. Finally, the same as the frame-based transformation of traditional DWT, the huge memory requirement is also a big problem for SA-DWT.

Here we emphasize on the second issue, which is the most significant problem for SA-DWT compared to traditional DWT. Fig. 3 shows the conditions that type B boundary extension is applied to several very short line segments.

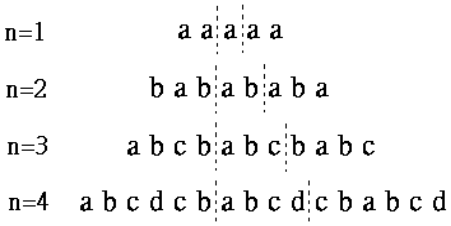


Figure 3: Type B Boundary extension for several very short line segments

If a traditional convolution-based DWT architecture is used for SA-DWT, then several problems are induced. One should use a finite state machine (FSM) controller with extremely high complexity and a large amount of multiplexers to maintain a regular data flow or do nothing but just introduce excess stalls for the boundary extension such that the data flow become irregular. Moreover, the ratio of required stalls to pixels becomes higher as the line segment becomes shorter. Assume the input samples are one even/odd pair in one clock cycle, Table 1 lists some overhead conditions induced by using traditional convolution-based DWT architecture for SA-DWT. In the worst case, the overhead ratio is even up to four.

Table 1: Some overhead conditions of traditional convolution-based DWT architecture for SA-DWT

Pixels	Stall Cycles
1	4
2	4
3	5
4	5
>= 5	6

2.2. The Lifting Scheme

The basic idea of lifting scheme is to factor the polyphase matrix of a wavelet filter into a series of upper and lower triangular matrices and a constant diagonal matrix [6][7]. Performing DWT with lifting scheme usually requires less multiplication and is easier to manage the boundary extension than performing DWT with convolution. The factoring algorithm can be extended from Euclidean algorithm for Laurent polynomials. By observing the factoring result, we can separate the wavelet transforms into a few temporary steps of multiplication and addition. One another important feature of lifting scheme is that the lifting steps of the synthesis filter can be easily obtained from that of the analysis filter. The polyphase matrix of a wavelet filter can be represented as

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

The key point of factoring the polyphase matrix is to make the degree of every Laurent polynomials, $s_i(z)$ and $t_i(z)$, as small as possible and keep symmetric or anti-symmetric for decreasing the number of required multiplication. Then we can construct a DWT with lifting scheme by the essential operation of lifting scheme, which consists of one multiplication and two additions as shown in Fig. 4. We call this operation the main computation unit (MCU).

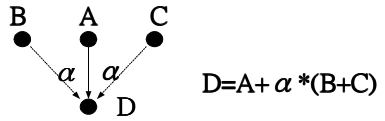


Figure 4: Essential operation of lifting scheme -- MCU

From Fig. 4, it is easy to see that how a DWT with lifting scheme can easily manage boundary extension. If the left point of a MCU is not an inside point according to the shape information, we can just add the right point of a MCU to itself instead of adding the left one. The case is similar if the right point is not an inside point. Therefore, implementing SA-DWT with lifting scheme can manage the boundary extension smoothly with few multiplexers and no excess stalls. In the following of this subsection, we will discuss the lifting scheme in detail and two odd symmetric biorthogonal filters are taken as examples.

2.2.1. (9,7) Odd Symmetric Biorthogonal Filter

The popular Daubechies (9,7) filter is the default of JPEG2000 lossy coding. Its wavelet filter coefficients is listed in Table 2.

Table 2: Daubechies (9,7) odd symmetric biorthogonal filter

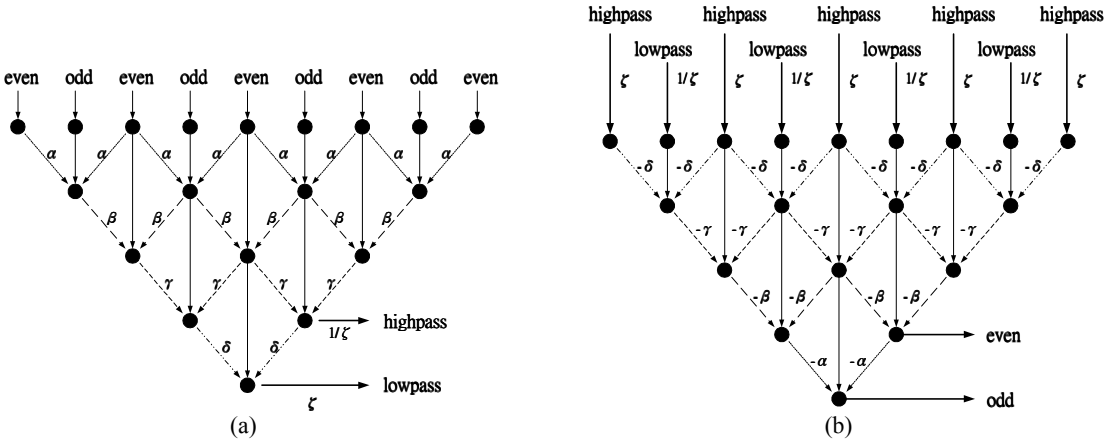
Low Pass	High Pass
0.6029490182363579	1.115087052456994
0.2668641184428723	-0.5912717631142470
-0.0782232665289878	-0.05754352622849957
-0.01686411844287495	0.09127176311424948
0.0267487574108097	

The (9,7) filter is factored into to a series of matrices that all have a symmetric Laurent polynomial of degree 1 as followed:

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

$$\alpha = -1.586134342;\quad \beta = -0.05298011854;$$
$$\gamma = 0.8829110762;\quad \delta = 0.4435068522;$$
$$\zeta = 1.149604398$$

We can interpret the lifting scheme of the analysis filter by the data flow in Fig. 5(a).



(a) (b)

Figure 5: (a) Forward (9,7) DWT with lifting scheme. (b) Inverse (9,7) DWT with lifting scheme.

In order to obtain the lifting scheme of the synthesis filter, we change the data flow as in Fig. 6 and show the result in Fig. 5(b).

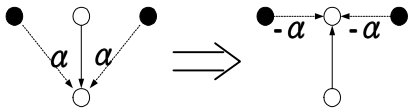


Figure 6: Changing the data flow

2.2.2. (9,3) Odd Symmetric Biorthogonal Filter

The Daubechies (9,3) filter is the default of MPEG-4 still texture coding. Its wavelet filter coefficients is listed in Table 3.

Table 3: Daubechies (9,3) odd symmetric biorthogonal filter

Low Pass	High Pass
0.99436891104360	0.70710678118655
0.41984465132952	-0.35355339059327
-0.17677669529665	
-0.06629126073624	
0.03314563036812	

The (9,3) filter is factored into two matrices with a constant diagonal matrix as followed:

$$\tilde{P}(z)=\begin{bmatrix}1 & \alpha(1+z^{-1})\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \beta(1+z)+\gamma(z^{2}+z^{-1}) & 1\end{bmatrix}\begin{bmatrix}\sqrt{2} & 0\\ 0 & \frac{1}{\sqrt{2}}\end{bmatrix}$$

$$\alpha = -0.5;\quad \beta = 0.296875;\quad \gamma = -0.046875$$

We can interpret the lifting scheme of the analysis filter by the data flow in Fig. 7(a). By changing the data flow as in Fig 6, the lifting scheme of the synthesis filter can be obtained as in Fig. 7(b).

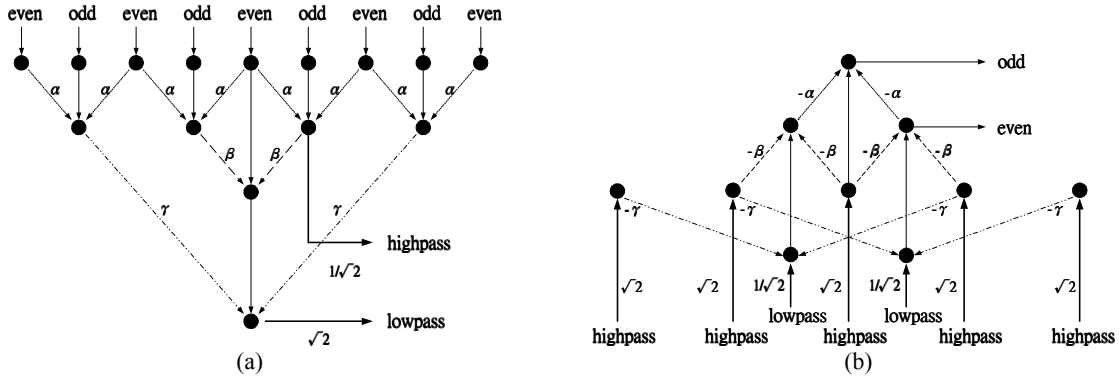


Figure 7: (a) Forward (9,3) DWT with lifting scheme. (b) Inverse (9,3) DWT with lifting scheme.

3. 1-D SA-DWT ARCHITECTURES

3.1. Generic Model of 1-D SA-DWT Architectures

As mentioned in subsection 2.1, there are four hardware implementation issues of SA-DWT. In order to overcome the first three, which are belonging to 1-D SA-DWT architectures, we proposed a generic model of 1-D SA-DWT architectures as shown in Fig. 8. In this generic model, there are four functional units. Functional unit A is a dedicated

architecture that can efficiently perform the subsampling operations and controlled by functional unit D. Functional unit B consists of several multiplexers controlled by signal sent form functional unit D. Functional unit C is the computational datapath corresponding to target wavelet filter. Functional unit D is the shape information analyzer that can analyze input shape information and send corresponding control signals to functional unit A and B. By this model, all critical problems of 1-D SA-DWT, including boundary extension, subsampling, and the requirement of filtering operations, can be solved in an easier and more economical way than convolution-based architecture. Any wavelet filter that is lifting factored and interpreted as data flow can be mapped into this model. In the following subsections, two wavelet filters that have been considered previously are mapped into their corresponding 1-D SA-DWT architectures.

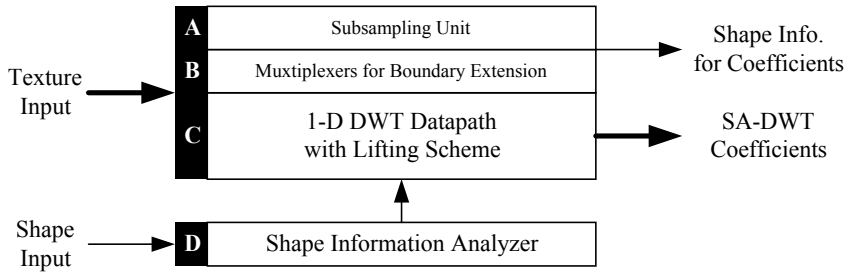


Figure 8: Generic model of 1-D SA-DWT architectures

3.2. 1-D SA-DWT Architectures of (9,7) filter

Since there are four computational stages among both forward and inverse transforms of the (9,7) filter, we can map every computational stage onto one MCU. The block diagram of 1-D forward SA-DWT architecture of the (9,7) filter is illustrated in Fig. 9. This architecture requires two input texture (even and odd) and shape data (evenmask and oddmask) and throws out two SA-DWT coefficients (lowpass and highpass) and shape information data (maskL, maskH, and e_or_o) in every clock cycle. We can easily obtain the block diagram of 1-D inverse SA-DWT architecture of the (9,7) filter from Fig. 9, because the main components in forward and inverse transforms are exactly the same. The most significant difference is the shape information analyzer. Besides, some differences would happen in the situation of one point line segment. Moreover, the sign of coefficients in multiplication is just the reverse of each other. Thus, we can combine these two 1-D architectures into a 1-D forward and inverse shared SA-DWT architecture by adding some multiplexers as shown in Fig. 10. Here we omit the detailed description of this architecture.

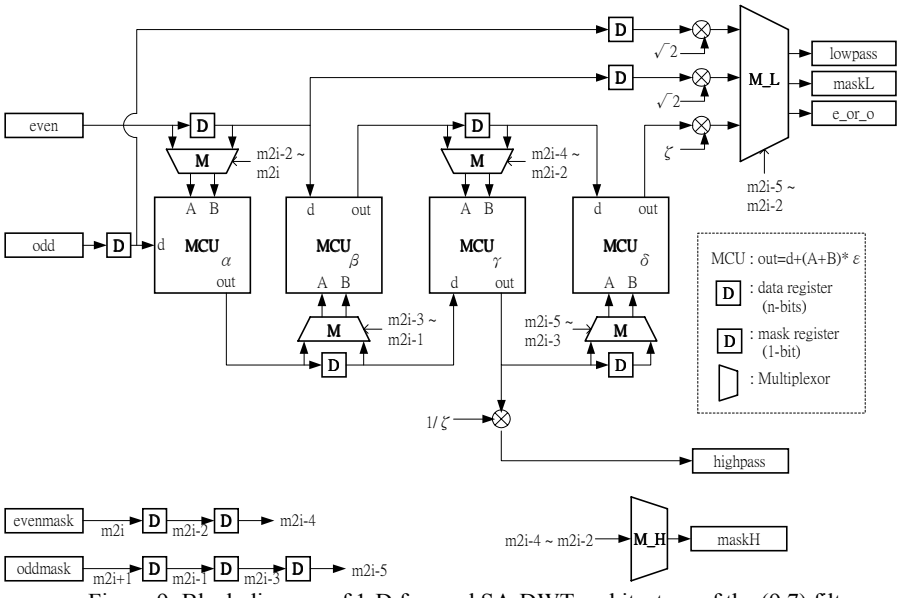


Figure 9: Block diagram of 1-D forward SA-DWT architecture of the (9,7) filter

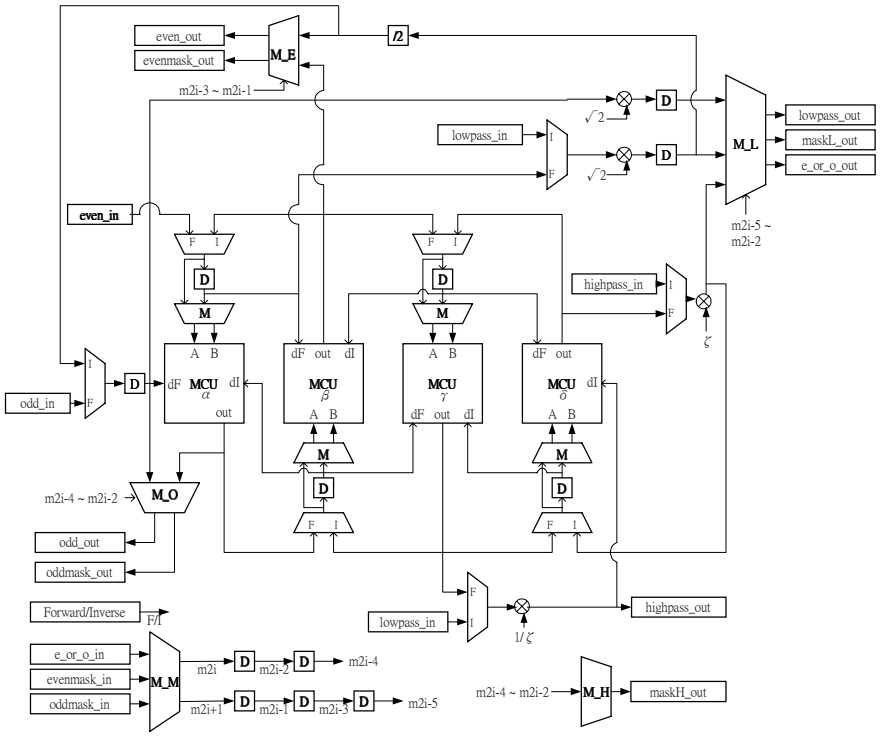


Figure 10: Shared 1-D SA-DWT architecture of the (9,7) filter

3.3. 1-D SA-DWT Architectures of (9,3) filter

The (9,3) filter can also be mapped into the generic model in the same way as the (9,7) filter. Here we show the shared architecture in Fig. 11. Since the shape information analyzer here is more complex than that of the (9,7) filter, a FSM controller is used inside the analyzer.

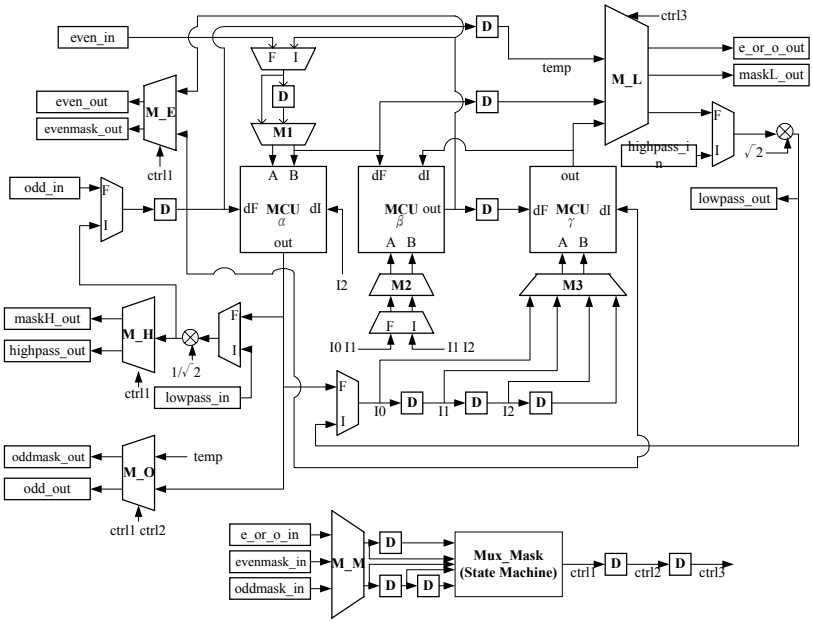


Figure 11: Shared 1-D SA-DWT architecture of the (9,3) filter

4. 2-D SA-DWT ARCHITECTURES

Another hardware implementation issue of SA-DWT or even all 2-D DWT architectures is the huge memory requirement. In this section, we would like to provide some high level insights for the implementation of 2-D DWT. Three architectures are discussed and some comparison results are given.

First, the most straightforward implementation of 2-D DWT is the direct method. That is, perform 1-D DWT in the row direction first and then in the column direction for each level. Observably, this architecture requires a frame memory which size is the same as the image. This huge memory requirement is usually the bottleneck of the hardware implementation. Moreover, we have to mention that the bandwidth of the frame memory access for J-level 2-D DWT decomposition is:

$$4\times(1+\frac{1}{4}+\frac{1}{16}+..+(\frac{1}{4})^{J-1})\times Nw\times Nh(pixels)$$

where *Nw* and *Nh* are the width and height of the image, respectively. The "4" in the above expression means the total memory access of the input and output for the two directions in every level.

Second, the priority of row and column direction is intrinsically the same. Therefore, it is unnecessary to always process row coefficients first. We suggest that the priority can be kept as row-column for the odd-level decomposition and reversed as column-row for the even-level decomposition. Thus, the consecutive row or column decompositions can be performed recursively in a Recursive Pyramid Algorithm [8] (RPA)-like architecture so as to reduce the frame memory access. We name it the RCCR architecture for its executive order of 2-D DWT is Row-Column-Column-Row. Except the first level, a half bandwidth of the frame memory access in other levels can be saved. The bandwidth of the frame memory access for J-level DWT decomposition in the RCCR architecture is:

$$[2+2\times(1+\frac{1}{4}+\frac{1}{16}+..+(\frac{1}{4})^{J-1})]\times Nw\times Nh(pixels)$$

Third, if no outside frame memory is expected or controllable, we can use the line-based method [9] to implement 2-D DWT. It performs row and column coefficients simultaneously, and throws out DWT coefficients immediately when they are not required in the following computation. Without the requirement of the frame buffer, line-based architecture needs some line buffers, which is proportional to *Nw*. Furthermore, the outside data access is the minimum, 2×*Nw*×*Nh* pixels.

In Table 4, we list some comparison results of above three architectures. About the system integration, we assume that there exists only one frame memory in a system. Thus, the direct method and the RCCR architecture need to occupy the frame memory during the processing time, which is roughly 2.5 times longer than that of the line-based architecture.

Table 4: Comparison of several 2-D architectures

Architecture	Least Memory Requirement	Outside Data Access (J→∞)	System Integration
Direct Method	O(*Nw*×*Nh*)	5.33	Difficult
RCCR	O(*Nw*×*Nh*)	4.67	Difficult
Line-based	O(*Nw*)	2	Easy

In the following of this section, we discuss two 2-D SA-DWT architectures, one uses the direct method and the other uses the line-based method. Their corresponding VLSI implementations are given in next section.

4.1. 2-D SA-DWT Architecture with Direct Method

Here we suppose that there is an external memory as the frame memory. In order to generate the read and write addresses for external frame memory, two address controllers are designed to calculate the related read and write addresses. The block diagram of complete 2-D SA-DWT architecture with direct method, including the shared 1-D SA-

DWT architecture, two address controllers, and the external frame memory, is shown in Fig. 12.

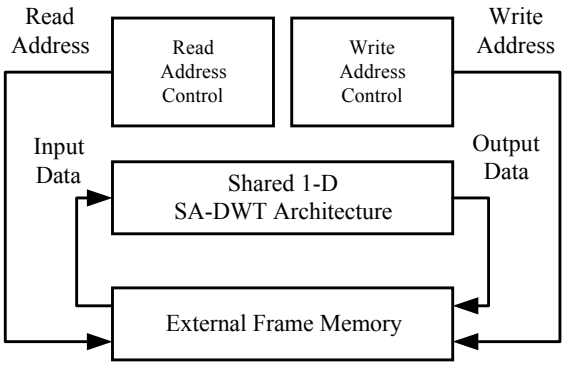


Figure 12: 2-D SA-DWT architecture with direct method

4.2. 2-D SA-DWT Architecture with Line-based Method

First of all, we assume that the raster scan format is used for the input of the forward SA-DWT and the output of the inverse SA-DWT. As shown in Fig. 13, two 1-D SA-DWT architectures (Row and Column SA-DWT) are used in proposed 2-D SA-DWT with line-based method. One is for the row direction processing, and the other is for the column direction processing. The registers in the column SA-DWT unit are replaced with the temporal buffer, which size is proportional to the width of image. On the other hand, the input data of the column SA-DWT unit are stored in the data buffer. The line buffer of this architecture consists of the temporal buffer and the data buffer. In the forward SA-DWT, the output data of the row SA-DWT unit are fed into the data buffer line by line and the column SA-DWT unit get intermediate coefficients immediately after they are available. The case of the inverse SA-DWT can be handled similarly. Therefore, we can find the minimum of the data buffer is $3\times Nw$. However, the size of the temporal buffer depends on which DWT filters are used and what kind of pipeline strategy is decided. If the 1-D SA-DWT has K data registers, the size of the temporal buffer would be $K\times Nw$.

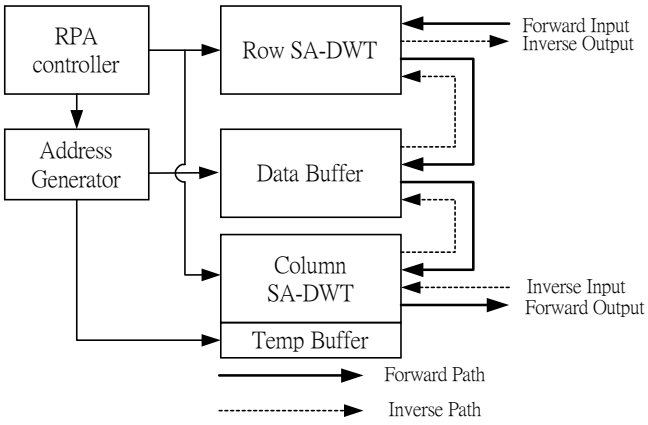


Figure 13: 2-D SA-DWT architecture with line-based method

Multi-level dyadic type SA-DWT decomposition can be achieved by using the RPA. Moreover, since the characteristic, two-read-two-write, of the 1-D SA-DWT architecture is difficult to be implemented by using the physical RAM, we modify the memory access requirement to one-read-one-write per clock by scheduling the RPA controller. We use an example to describe the RPA schedule and the general case can be derived easily. The example for a 3-level decomposition is given in Fig. 14. It should be noted that the amount of coefficients in the LL band is 1/4 of that in the original image. For multi-level decomposition, the size of the data buffer is constant for any kind of DWT filters and can be calculated as $3\times Nw\times(1+1/2+\ldots+(1/2)^{J-1})$, where J is the decomposition level. However, the size of the temporal buffer is proportional to the number of data registers K in the 1-D SA-DWT architecture and can be calculated as

$K \times Nw \times (1+1/2+\ldots+(1/2)^{J-1})$.

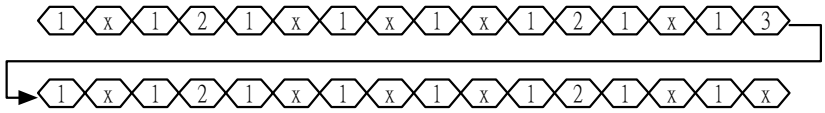


Figure 14: RPA schedule for 3-level 2-D DWT decomposition

5. VLSI IMPLEMENTATIONS

5.1. VLSI Implementation of 2-D SA-DWT with Direct Method

In the first VLSI implementation, a prototyping core layout of proposed 2-D SA-DWT with direct method using Daubechies (9,7) filter has been implemented by using cell-based chip design flow. The core layout features are listed in Table 5. The datapath precision of this design is 12-bit coefficient by 16-bit data for multiplier and 16-bit data for adder. This design occupies 1455 by 1455 um^2 core size under TSMC 0.35um CMOS 1P4M process. The maximum clock rate is 50MHz, and for 1024 by 1024 frame size, 36.3 frame/sec with 3-level decomposition can be achieved. Since we use an external memory as the frame memory, the bandwidth of this memory is 50MHz with two-read and two-write 18-bit data access.

Table 5: Features of the core layout of 2-D SA-DWT with direct method

Technology	TSMC 0.35um CMOS 1P4M
Core Size	1455 x 1455 um^2
Transistor Count	104269
Max Clock Rate	50MHz
Wavelet Filter Bank	(9,7) Biorthogonal
Decomposition Level	3
Frame Size/Rate (Extendable)	256x256 (581.2 frames/sec)
	512x512 (145.3 frames/sec)
	1024x1024 (36.3 frames/sec)
External Memory BW	50MHz x 2R2W x 18-bit

An experimental result of the test sequence Akyio is shown in Fig. 15. Fig. 15(a) is the original visual texture, Fig. 15(b) is the related shape information, Fig. 15(c) is the 3-level forward transform result, and Fig. 15(d) is the reconstructed visual texture after inverse transform.

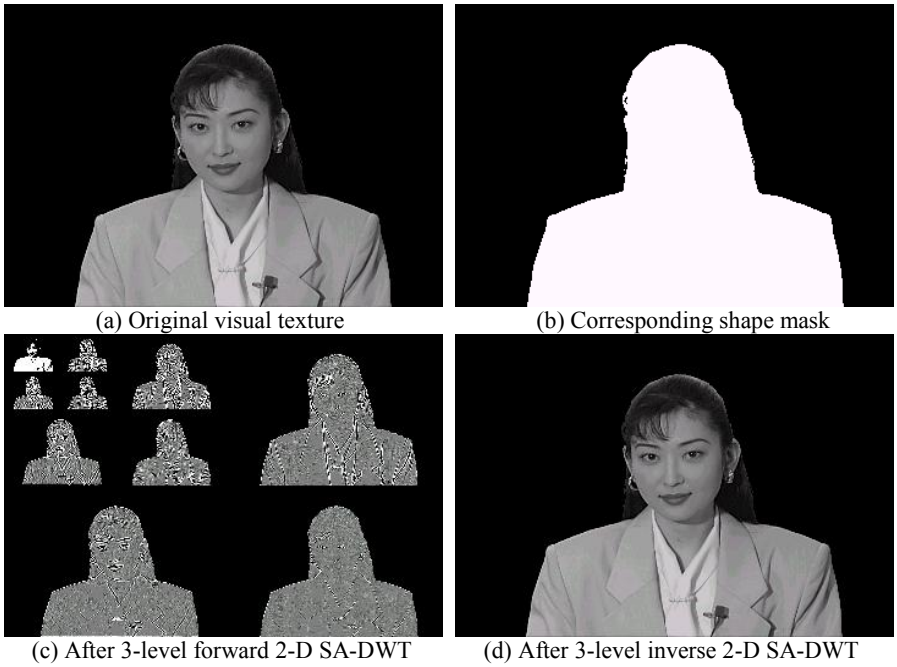
5.2. VLSI Implementation of 2-D SA-DWT with Line-based Method

In this implementation, a 3-level 2-D SA-DWT architecture with line-based method using the Daubechies (9,3) filter is realized. Except the normalization step, we use shifts and additions to replace the integer multiplications in the lifting scheme of the (9,3) filter. Furthermore, we exploit the characteristic of the 2-D DWT decomposition and the normalization coefficients, which are $\sqrt{2}$ and $\sqrt{2}^{-1}$, to replace the normalization with shift-right and shift-left. Therefore, no multipliers are required in the implementation of the (9,3) filter. A prototyping chip of this implementation has been fabricated by TSMC. The features of this chip are listed in Table 6, and Fig. 16 shows the layout of this chip.

6. CONCLUSION

In this paper, we have presented several VLSI architectures and implementations of SA-DWT with odd symmetric biorthogonal filters. Based on the analysis results of SA-DWT algorithm, 1-D SA-DWT architectures, which consists of several dedicated shape processing units and a lifting-based DWT datapath, are proposed for 1-D length-adaptive DWT.

Using the 1-D SA-DWT architectures as basis, two 2-D SA-DWT architectures are proposed and VLSI implemented. The presented architectures can be applied both in arbitrarily shaped visual objects coding such as MPEG-4 still texture coding and rectangular region image coding such as JPEG2000 still image coding.



(a) Original visual texture (b) Corresponding shape mask

(c) After 3-level forward 2-D SA-DWT (d) After 3-level inverse 2-D SA-DWT

Figure 15: An experimental result of the test sequence Akyio

Table 6: Features of the 2-D line-based SA-DWT chip

Technology	TSMC 0.35um CMOS 1P4M
Package	100 CQFP (89 Pads used)
Die Size	3.70 x 4.11 mm^2
Core Size	2.88 x 3.29 mm^2
Transistor Count	535848
Max Clock Rate	33MHz
Wavelet Filter Fank	(9,3) Biorthogonal
Decomposition Level	3
On-Chip Memory	11 112x16 two port RAMs for temp buffer
	6 64x18 two port RAMs for data buffer
	Total 26624-bit (25424-bit used)
Power Consumption	340mW @ 3.3V, 33MHz

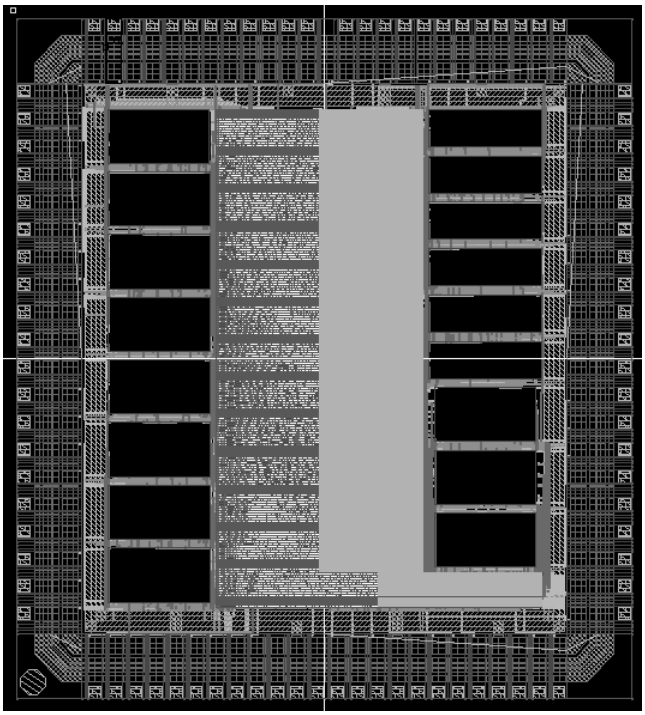


Figure 16: Layout view of the 2-D line-based SA-DWT chip

REFERENCE

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

[2] ISO/IEC, ISO/IEC 15444-1, Information technology – JPEG2000, image coding system, 2000.

[3] MPEG-4 Video Group. Generic Coding of Audio-Visual Objects: Part 2 – Visual, ISO/IEC JTC1/SC29/WG11 N2802, FPDAM1 of ISO/IEC 14496-2, Atlantic City, July 1999.

[4] S. Li and W. Li, "Shape-adaptive discrete wavelet transforms for arbitrary shaped visual object coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 5, pp. 725-743, Aug. 2000.

[5] G. Xing, J. Li, S. Li and Y. Zhang, "Arbitrary shaped video object coding by wavelet," *IEEE International Symposium on Circuits and Systems*, III-535 ~ III-538, May 2000.

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

[7] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting schemes," *The Journal of Fourier Analysis and Applications*, Vol. 4, p247-269, 1998.

[8] 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.

[9] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Transactions on Image Processing*, Vol. 9, No. 3, March 2000.