

AN EMBEDDED EXTENSION ALGORITHM FOR THE LIFTING BASED DISCRETE WAVELET TRANSFORM IN JPEG2000

K. C. B. Tan and T. Arslan

Department of Electronics & Electrical Engineering,
The University of Edinburgh,
The King's Buildings, Mayfield Road,
Edinburgh EH9 3JL, Scotland, UK
Phone: +44 131 650 5619 FAX: +44 131 650 6554
Email: benny.tan@ee.ed.ac.uk

ABSTRACT

This paper presents a novel algorithmic technique for reducing power and area consumption of the Discrete Wavelet Transform (DWT) when implemented in VLSI hardware. The technique reduces power by combining the data-extension procedure into the lifting-based DWT core resulting in a significant reduction in the amount of memory requirements and associated read/write operation together with a reduction in the number of arithmetic operations. This in turn has the effect of reducing the amount of switched capacitance within the hardware unit. We demonstrate that the new algorithm can be used to obtain more than 50% reduction in the amount of memory required leading to significant reduction in area and power.

1. INTRODUCTION

The Discrete Wavelet Transform (DWT) is increasingly being used for image coding. This is because it has features such as progressive image transmission by quality/resolution, and ease of compressed image manipulation. The above features have also lead to significant interest in efficient algorithms for hardware realisation [1] of the DWT. The conventional convolution based DWT is computationally intensive and both area and power hungry. Some of these drawbacks were overcome by using the *lifting based scheme* for the DWT [2,3], which has been selected in the recently released JPEG2000 [4] standard. Such developments aroused considerable interest in the implementation [5,6] of this algorithm. With the increase in demand for video telephony, embedded low power video becomes an

essential feature. Low power video is important in such cases, as reasonable operating time, due to the high power consumption of video application, cannot be met by current battery technology. So far, only a few researchers have tackled the low power implementation of the DWT, through *generic algorithmic* techniques. Even with the lifting based scheme, few have targeted their work on low power implementation of this algorithm, especially through the reduction of switched capacitance [7] by reducing computation steps. Such implementation approaches are desirable since they yield in generic circuits with no added design effort. Most research work to date has considered the implementation of the DWT targeting reducing the computational complexity of the design and modifying it to operate on low supply voltage. For example the work in [8] targets reducing the power consumption of the DWT focusing on the modification of certain sections in order to operate under a lower supply voltage, which requires additional design effort to compensate for the added delay.

This paper presents a novel algorithmic technique for reducing the power dissipation and memory requirements for data-extension prior to one-dimensional DWT. As DWT is implemented as a Finite Impulse Response (FIR) filter, data-extension is important to ensure that the FIR filter, at the edge/boundary of an image, has enough data in order to produce a single output. Power reduction is achieved by combining and embedding data-extension into the main DWT algorithm in both the start and the end of the transformation process. This reduction is achieved as a result of the following; (i) reduction in the amount of storage memory since explicit extensions are not required, (ii) reduction in the number of read/write accesses since extension is only performed when the main data term is read or written, (iii) reduction in the number of arithmetic operations (see

later). The data-extension used here is *symmetrical* (see figure 1) as specified by JPEG2000 standard. The JPEG2000's 5/3 DWT (with lifting) is used to demonstrate this technique.

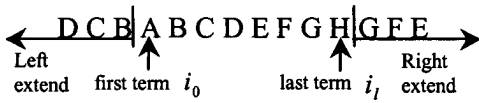


Figure 1: Symmetrical data extension

2. ALGORITHM

Conventional straightforward data-extension involves extending the data to the right, copying all the original data and lastly performing a left extension on the data (see figure 2).

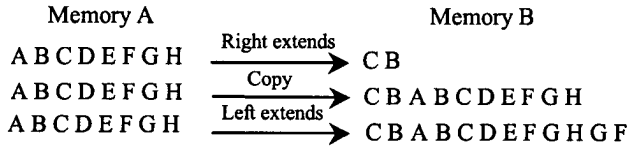


Figure 2: Conventional straightforward data-extension

The number of data terms to extend will depend on whether the data set starts or ends with an even or odd number indexed term. If the data starts or ends with an even number indexed term, then two data terms are extended to the left and the right of the data set (see figure 3 & 4). If the data starts or ends with an odd number indexed term, then a single data extension is added to both the left and the right of the data set (see figure 3 & 4).

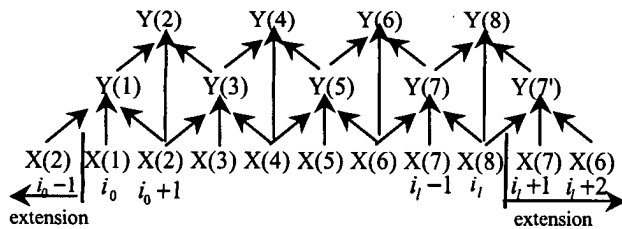


Figure 3: Odd start term and even end term data

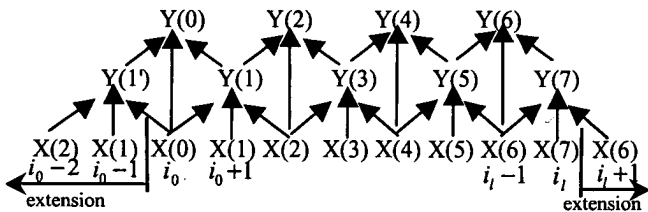


Figure 4: Even start term and odd end term data

Equations (1) and (2) describe the implementation of the 5/3 lifted forward DWT. Equation (1) is used for the computation of the odd coefficients, whereas equation (2) is used for the computation of the even coefficients. The computations of the odd coefficients have to be performed first followed by the computations of the even coefficient.

$$Y(2n+1) = X(2n+1) - \left[\frac{X(2n) + X(2n+2)}{2} \right] \quad (1)$$

$$Y(2n) = X(2n) + \left[\frac{Y(2n-1) + Y(2n+1) + 2}{4} \right] \quad (2)$$

Figures 3 and 4 depict the relationship between the output $Y(n)$ with its extended input $X(n)$. Our embedded extension algorithm splits the homogenous conventional computation process into three parts. The first part we term the *Start Process*, the second part the *Normal Process* (which is the same as the conventional computation), and the third part the *End Process*. The data-extensions are only embedded during the *Start Process* and the *End Process*. The *Start Process* embeds the left extension and the *End Process* the right extension. There are two equations for computing the coefficients in the *Start Process*. The first, equation (3), is the embedded extension equation for the computation of the first odd coefficient when the input data starts with an odd number indexed term. The second, equation (4), is used for the computation of the first even coefficient when the input data starts with even number indexed term. Note that i_0 in $Y(i_0)$ indexes the first term of the data.

$$Y(i_0) = X(i_0) - X(i_0 + 1) \quad (3)$$

$$Y(i_0) = X(i_0) + \left[\frac{Y(i_0 + 1) + 1}{2} \right] \quad (4)$$

As $X(i_0 - 1)$ is equal to $X(i_0 + 1)$ (see figure 3), equation (1) will reduce to equation (3). Also, using the fact that $Y(i_0 - 1)$ is equal to $Y(i_0 + 1)$ (see figure 4), equation (2) reduces to equation (4). Similarly in the *End Process*, equation (1) and (2) reduce to equation (5) and (6) respectively. Equation (5) and (6) are the embedded extension equation for the computation of the last odd coefficient, when the input data ends with an odd number indexed term (see figure 4), and the last even coefficient when the input data ends with even number indexed term (see figure 3) respectively. The i_l in $Y(i_l)$ indexes the last term of the data.

$$Y(i_t) = X(i_t) - X(i_t - 1) \quad (5)$$

$$Y(i_t) = X(i_t) + \left[\frac{Y(i_t - 1) + 1}{2} \right] \quad (6)$$

As could be deduced from the new set of equations (3-6), using this technique, computation steps are reduced as the entire computation of the odd coefficient is omitted when there is either an even number indexed start or end terms. In the same effect, a read access and two arithmetic operations are omitted in computing the first/last odd coefficient when there is either an odd number indexed start or end term. This technique can be extended to be used in the inverse DWT with the 5/3 filter and/or in both the forward and inverse DWT with a 9/7 filter.

3. RESULTS

Three techniques were evaluated in a one-dimensional access with a 5/3 DWT filter. These are; the conventional straight-forward extension (Conventional), extension by scheduled access (Scheduled access) and our new embedded extension algorithm (Embedded). The extension by scheduled access is a scheme where the data (read) access controller is scheduled to read the original data in a manner that the data are extended systematically, so that the DWT processor can calculate all the coefficients in a homogenous process. Analysis of equations (3-6) clearly points to both memory and power savings as could be seen from the last column of table 1.

Technique	Case	Mem. Used	Mem. Rd.	Ext. no. of Ops.	Ops. Rd.
Conventional	a	N+2	0	2N+4	0
	b	N+3	0	2N+6	0
	c	N+4	0	2N+8	0
Scheduled access	a	-	N+2	2	2N+2
	b	-	N+3	3	2N+3
	c	-	N+4	4	2N+4
Embedded	a	-	N+2	-2	2N+6
	b	-	N+3	-5	2N+11
	c	-	N+4	-8	2N+16

Table 1: Memory and power (as number of operation) savings per single processed row/column

Note that 'N' in Table 1 represents the number of data in a row or column. 'Extension access number of operations' (Ext. no. of ops) refers to read/write access operations and 'Operation reduction' (Ops Rd.) refers to the reduction of the operations in the extension process of the each respective scheme. 'Memory Used' (Mem. Used) refers to the additional memory used in the extension operation and lastly 'Memory Reduction' (Mem. Rd.) refers to the memory reduction of the each respective scheme. Three cases of extension are considered in the calculation of computational and memory savings. In the first case (*case a*), we consider the extension with input data that starts with an odd number indexed term and ends with odd number indexed term. The second case (*case b*) considers input data that either starts with an even number indexed term and ends with odd number indexed term or starts with odd number indexed term and ends with even number indexed term. The third case (*case c*) considers extensions with input data that starts with even number indexed terms and ends with even number indexed term. The output of our embedded algorithm is verified and compared to that of the conventional straight-forward extension using MATLAB. In the conventional straight-forward extension, each data term will incur a two operation overhead with a read access and write access. In extension by scheduled access, each data in the extension only incur one operation. Our new embedded extension algorithm does not incur any extension overhead. Besides this, our algorithm also saves up to an additional of eight arithmetic operations (for data that starts and ends with even number indexed term) per row/column, as several computations are eliminated as shown above. Although the savings with our embedded extension algorithm in one-dimensional process are only a few operations more than those of extension by scheduled access, savings will be considerable when it comes to two-dimensional processing of a whole frame. As the table shows, significant savings in memory and number of operation saving are achieved. Such major savings lead to significant savings in both area and power. The algorithm has a slight overhead in terms of a small increase in the area of the control logic. This is mainly due to incorporating Start/End states into the controller. Although there is an increase in control area, the overall reduction in power consumption will not be affected. This is because the increase is insignificant compare to the savings achieved, due to the reduction in memory size and the number of arithmetic operations. Furthermore, the additional states can be effectively manipulated by appropriate power-down mechanisms.

4. CONCLUSIONS

An algorithm has been presented which results in more efficient hardware realization of the Discrete Wavelet Transform. We have shown that the algorithm can lead to hardware architectures with significant reduction in power and area. The reduction is achieved by combining the data-extension into the main lifted-based DWT such that the computation steps are reduced. We have shown that the algorithm can be used to obtain more than 50% memory and power reduction.

Aknowledgement

This work is supported by the Engineering and Physical Sciences Research Council under grant number GR/N08322.

5. REFERENCES

- [1] C.Yu and S-J. Chen, 'VLSI implementation of 2D discrete wavelet transform for real-time video signal processing', *IEEE Transactions on Consumer Electronics*, Vol. 43, No. 4, Nov 1997.
- [2] W. Sweldens, 'The lifting scheme: A new philosophy in biorthogonal wavelet constructions', *Proceedings of SPIE*, 2569, pp.68-79, 1995.
- [3] I. Daubechies and W. Sweldens, 'Factoring wavelet transforms into lifting steps', *J. Fourier Analysis and Applications*, Vol. 4, pp.247-269, 1998.
- [4] *ISO/IEC, ISO/IEC15444-1, Information technology - JPEG 2000 image coding system*, 2000.
- [5] K. Andra, C. Chakrabarti and T. Acharya, 'A VLSI architecture for lifting-based wavelet transform', *2000 IEEE Workshop on Signal Proceeding Systems*, SiPs 2000, pp.70-79, Oct 2000.
- [6] C-J. Lian, K-F. Chen, H-H.Chen and L-G. Chen, 'Lifting based discrete wavelet transform architecture for JPEG2000', *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems*, ISCAS 2001, Vol. 2, pp.445-448, 2001.
- [7] S. Masupe and T. Arslan, 'Low Power DCT Implementation Approach for CMOS-based DSP Processors', *IEE Electronics Letters*, Vol. 34, No. 25, pp.2392-2394, Dec 1998.
- [8] T. Simon and A. P. Chandrakasan, 'An ultra low power adaptive wavelet video encoder with integrated memory', *IEEE Journal of Solid-State Circuits*, Vol. 35, No.4, Apr 2000.