

SHIFT-ACCUMULATOR ALU CENTRIC JPEG2000 5/3 LIFTING BASED DISCRETE WAVELET TRANSFORM ARCHITECTURE

K-C. B. Tan and T. Arslan

Institute of Micro and Nano System,
School of Engineering and Electronics, The University of Edinburgh,
The King's Buildings, Mayfield Road, Edinburgh EH9 3JL, Scotland, U.K.
Phone: +44 131 650 5619 Fax: +44 131 650 6554
Email: Benny.Tan@ee.ed.ac.uk

ABSTRACT

This paper presents a novel (low arithmetic unit count) hardware architecture for performing lifting-based JPEG2000's 5/3 Discrete Wavelet Transform (DWT). The architecture is built around parallel Shift-Accumulator Arithmetic Logic Units (ALUs) which can encode (with implicit embedded extension[8]) up to five levels of transformation. The proposed architecture, which consists of three adders, two subtractor-adders and five shifters, has a significantly lower hardware count compared to the architectures proposed by K. Andra et. al. [5] and C.-J. Lian et. al. [6]. In addition, the architecture has an efficient memory organisation, which uses lesser amount of embedded memory for processing and buffering. In this paper, we present the architecture and demonstrate that it closely adheres to the JPEG2000's specification while reducing the hardware requirements and hence area and power consumption.

1. INTRODUCTION

Discrete Wavelet Transform (DWT) is becoming increasingly popular 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-mentioned features have lead to significant interest in efficient algorithms for hardware realisation [1] of the DWT. 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 released JPEG2000 [4] standard. Such developments aroused considerable interest in the implementation [5][6][7] of this algorithm.

With the increase in demand for portable 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 JPEG2000's DWT. Even with the lifting based scheme, few have targeted their work on low power implementation of this algorithm, especially through a reduction of switched capacitance [10] by reducing the number of computational steps and datapath/arithmetic hardware. Most low power research work to date has considered the implementation of the DWT by targeting to reduce the computational complexity of the design and

modifying it to operate on low supply voltage. For example the work in [11] 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. Recent implementation [5][6] of the JPEG2000's DWT are pipeline-datapath centric designs, which have disregarded the issue of power altogether. Furthermore, these works assumed a horizontal (row) feed and process first which does not comply with JPEG2000's specification. Such architecture either assumes that the whole image frame is available before the processing starts or requires buffering of the whole or part [5] of the frame.

In this paper, we present a novel shift-accumulator (SA)-ALU centric lifting-based scheme 5/3 DWT processor architecture, which has significantly lower hardware count, in terms of arithmetic and memory units compared to architecture presented in [5][6]. With its five level dyadic addressed and structured memory buffers design, this architecture has the capability of performing multi-level DWT encoding up to five levels (at one level at a time). With a slight modification made to its control by either software or hardware, this architecture can also be used as a decoder. Depending on the orientation of image vectors that are fed into its input, this system could start its computation in either the row or column direction. However, in this paper we assumed the transformation process is in the column-row fashion. The architecture is made up of two sets of processors and two sets of buffering memory units. The first processor consists of two vertical (column) SA-ALUs and the second processor consists of three horizontal (row) SA-ALUs. The two sets of memory buffers are used to store processed output data, which are either for further processing at the next level or for the subsequent row processing.

This paper is organised as follows. In Section 2, we will briefly describe the lifting scheme DWT and JPEG2000's 5/3 DWT. Then, the proposed architecture and the organisation of its components and their workings will be presented in Section 3. This is followed by Section 4, which presents the details of how the architecture operates. Next, in Section 5, the proposed architecture will be compared with existing architectures. And finally, we conclude this paper in Section 6.

2. LIFTING-BASED DWT

Conventional convolution-based DWT (figure 1a) is computationally intensive and area wasting, which makes it

power hungry. This is because conventional DWT wastes redundant computing power and memory space on processing and storing data vectors, which half of these will be dropped during the down-sampling later on in the process. These drawbacks are overcome by the lifting-based DWT (figure 1b), making hardware implementation of DWT viable. The lifting-based DWT reduces computational and area redundancy by: - Firstly, taking into account the redundancy of down-sampling and avoiding computation of the output vectors that will eventually be dropped. Secondly, exploiting the similarities between the low-pass (LPF) and high-pass filter (HPF) to further reduce redundancies.

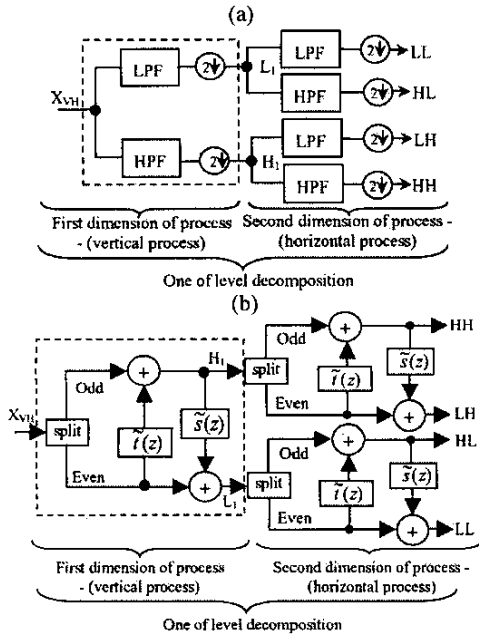


Figure 1. 2D DWT: a) Conventional convolution-based b) Lifting-based

In order to utilise the similarities, common factors of both filters must first be extracted into lifting steps [2][3] which consists of three distinct steps. The three steps are 1) *predict step* ($\tilde{i}(z)$), 2) *update step* ($\tilde{s}(z)$) and 3) *scaling step* (K) (K is a constant which is not shown in the figure 1b). With these steps, a set of (lifting-based) equations is obtained. These equations are then used to compute the final down-sampled outputs. If LPF and HPF are *complementary*, then the transform is reversible (i.e. perfect reconstruction is possible).

$$(a) \quad HPF = Y_h(n) = -\frac{1}{2}X_{ev}(n-1) + X_{ev}(n) - \frac{1}{2}X_{ev}(n+1)$$

$$(b) \quad LPF = Y_l(n) = -\frac{1}{8}X_{ev}(n-2) + \frac{2}{8}X_{ev}(n-1) + \frac{6}{8}X_{ev}(n) + \frac{2}{8}X_{ev}(n+1) - \frac{1}{8}X_{ev}(n+2)$$

Equation set 1. Transfer functions of 5/3 DWT analysis filters: a) HPF b) LPF

The original 5/3 DWT transfer function (see equation set 1) can be factorised in order to obtain equation set 2. It is noteworthy that the outputs of the lifting-based 5/3 DWT are indexed by either odd or even terms.

$$(a) \quad H = Y(2n+1) = X_{ev}(2n+1) - \left[\frac{X_{ev}(2n) + X_{ev}(2n+2)}{2} \right] \quad \begin{array}{l} \text{Odd terms} \\ \text{(high pass} \\ \text{output)} \end{array}$$

$$(b) \quad L = Y(2n) = X_{ev}(2n) + \left[\frac{Y(2n-1) + Y(2n+1) + 2}{4} \right] \quad \begin{array}{l} \text{Even terms} \\ \text{(low pass} \\ \text{output)} \end{array} **$$

** JPEG2000's 5/3 even terms have an additional term of 0.5 added to it.

Equation set 2. Lifting-based 5/3 forward (analysis) DWT: a) Odd terms (HPF) b) Even terms (LPF)

From figure 1b and equation set 2, it can be seen that the lifting-based scheme reduces more than 50% of the conventional DWT computational requirement. Even with this significant reduction, there is still room for further reduction in computation and area in the implementation of a lifting-based DWT system. By careful design, redundancies can be further reduced.

3. PROPOSED ARCHITECTURE

The proposed architecture in figure 2 computes the data vectors in 2-dimension (2D) (with implicit embedded extension [8]) in the column-row fashion. It can be seen from the figure that the 2D lifting-based DWT is mapped into two main processors: - 1) the Column Processor and the 2) the Row Processor.

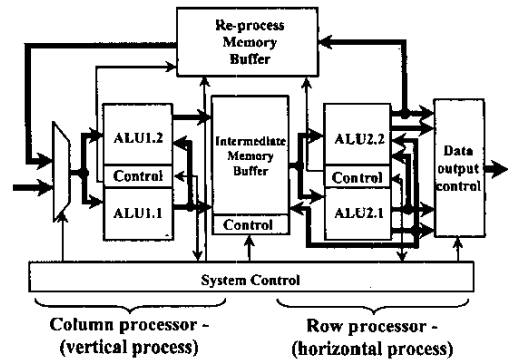


Figure 2. Overall block diagram of the proposed architecture

The architecture consists of the following 3 main components: - 1) Control Unit (CON) 2) SA-ALUs and 3) Buffering Memory (MEM). Each of the main components depicted in figure 2 has its own control. All of these individual controls are co-ordinated and synchronised with one another via the system control. Each processor has its own control which decides when and how many right shifts are to be applied to the chosen vectors. These control units also control all the steps of the rest of the arithmetic operations. There are a total of five SA-ALUs in the architecture; two in the column processor and three in the row processor. ALU1.1 and ALU2.1 are odd term ALUs, whereas ALU1.2 and ALU2.2 are even term ALUs. This architecture is a line-based DWT [9] architecture, therefore it does not require frame buffers. However, there are two major blocks of memory components in the architecture. These are the Intermediate Memory Buffer (INMEM) and the Re-process Memory Buffer (RPMEM). Intermediate data vectors from both the column processor and the odd outputs of row processor are stored in the INMEM. RPMEM is used to store LL outputs from the row processor for the next level of analysis process. Details of the SA-ALU and the make

up of the memory buffers will be presented in the following subsections.

3.1 SA-ALU Architecture

As the lifting-based 5/3 DWT coefficients are factors of base two ($\frac{1}{2}$ and $\frac{1}{4}$ - as seen from equation 2), processors consisting of shifter and adder are sufficient to realise the transform. For this reason, all ALUs in this architecture are SA-ALUs. All SA-ALUs in the architecture are based on (or a slight variation of) the ALU architecture shown in figure 3. The SA-ALU consists of the following: 1) input multiplexer, 2) shifter unit, 3) adder or subtractor-adder, 4) accumulator registers and 5) output and feedback multiplexer.

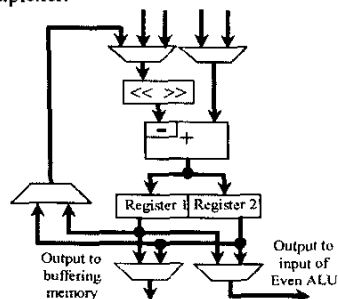


Figure 3. Simplified basic architecture of shift-accumulator ALU

The input multiplexers are used to switch the appropriate data (by the processor's control) from three data buffer registers to the ALU. The shifter realises the lifting-based 5/3 DWT coefficients by applying one right shift for the coefficient of $\frac{1}{2}$ and two right shifts for the coefficient of $\frac{1}{4}$. Odd term ALUs have a subtractor-adder each, whereas even term ALUs only utilise adder. Except for ALU2.2, which has two SA-ALUs, all the rest of the ALU units have only one unit of SA-ALU internally. The feedback multiplexer is used to select the data in the accumulators for further processing or accumulation. Output multiplexers are used to select final accumulated results in either of the accumulator registers for the INMEM and the inputs of the even term ALU.

3.2 Memory Architecture

Both memory buffers utilised are dyadic addressed dual-port memory modules. The RPMEM is implemented with a First-In-First-Out (FIFO) memory module. The data input port of the RPMEM is connected to the LL band output port of the row processor (ALU2) and the data output port is connected to the multiplexer input of the column processor (ALU1). ALU2 control unit supplies the write signals and ALU1 control unit supplies the read signals. The size of this re-process memory is $\frac{N}{2} \times \frac{N}{2}$, where N is the maximum number of image pixel data terms per row/column of a single image frame.

The intermediate memory buffers consist of six individual (one port write, one port read) dyadic addressed dual-port memory blocks (see figure 4). The six dual-port memory blocks are

grouped into two main blocks, an odd block and an even block. The two main blocks are each further divided into three columns; column 0, column1 and column 2. OC0, OC1 and OC2 are the output data ports of the odd term blocks of column 0, 1 and 2 respectively. Similarly, EC0, EC1 and EC2 are the output data ports of the even term blocks of column 0, 1 and 2 respectively. All of the six memory blocks are connected to the same write and read address lines. The write and read address lines of all the memory blocks are connected to the same write address and read address generator respectively. By sharing these address lines, hardware and power needed for addressing are reduced. Write enable controls of each of the memory blocks are generated separately and fed into their respective memory blocks. Each of these memory blocks is selected for writing by activating the enable signals (Ctrl o0 to o2 and Ctrl e0 to e2). There is no read control signal for the read ports as these are always read enabled and switched by ALU2 control unit via the input multiplexer of ALU2.

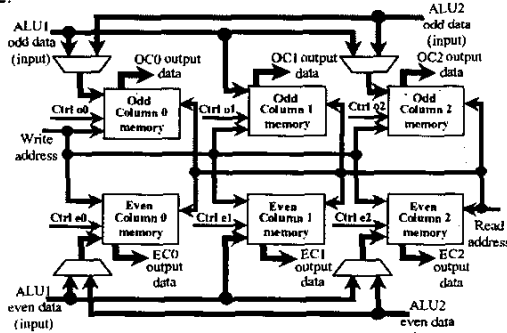


Figure 4. Intermediate memory buffer architecture

These memory buffers are designed to store up to three columns (i.e. 3N) of processed data. As the intermediate data from ALU2 need only to be stored into column 0 and 2, data from ALU1 and ALU2 are multiplexed into the write data ports of column 0 and 2 memory buffers via the data multiplexers. The remaining write data ports of column 1 are connected only to ALU1. Read data ports of the memory blocks are individually connected to the input multiplexer of ALU2. The total size of the intermediate memory is 3N.

4. OPERATION SCHEDULE

The architecture operation schedule was generated by first listing an efficient and detailed schedule of all the column processor's operations by hand. Then, using this column processor's schedule as a timing reference, the schedule of the row processor was drawn out. At the beginning of each frame, the process first starts with the column processor. The output data from the column processor are stored into the INMEM in a column-wise (to-and-fro) fashion. When a column (both odd and even) of memory block are filled, the next adjacent column of memory block will be used to store the subsequent streams of data. The processor will first fill-up column 0 then column 1. After that, it will fill-up column 2 before filling up column 1 again. Henceforth, the storing process repeats itself with column 0 as above.

The row processor will only commence its (column-wise) operation when the column processor starts processing the third column data. This is because it needs at least three input vector

terms to calculate an odd terms (see equation set 2a). After commencing the horizontal process, the row processor will stop processing when it reaches the end of the third column. It will idle and wait while the column processor processes the next column and stores its outputs into column 1 memory blocks. The row processor will resume processing when the column processor starts computation on the next subsequent column, storing its outputs into the even numbered INMEM column (i.e. 0 or 2). Henceforth, this process will repeat itself with the row processor only starting its (column-wise) processing when the column processor starts computation on even numbered (i.e. 0 or 2) columns. The row processor will only stop when the column processor is processing and storing data into column 1. When the row processor is processing with data from column 2, it stores its intermediate data back into column 0 and vice versa. When the row processor is running, it also outputs four sets of data over a period of three clock cycles. These outputs are fed into the data output control module, the RPMEM and the INMEM. The row processor will use the data previously stored in the intermediate memory in subsequent column processing.

5. RESULTS AND ANALYSIS

From section 3, it can be seen that the total amount of adders/shifters and registers in the 2D architecture is 5 and 10 respectively. Table 1 provides a comparison of our architecture with most recent work in the literature. The N in the table represents the maximum number of elements in a row/column of the input data.

	K. Andra et. al [5]	C-J. Lian et. al [6]	Our proposed architecture	Worst case % saving
Adders	8	8	5	37.5%
Shifters	4	4	5	-25%
Registers	16*	16	10	37.5%
INMEM	4N	NXN	3N	25%
RPMEM	$N \times \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$	0%

* Excluding register files

Table 1. Comparison of hardware counts

It can be seen from table 1 that our proposed architecture has a significantly lower number of hardware components in comparison to the other two referenced architectures. Hardware savings in all of the listed components (except for shifter and reprocess memory) are achieved. The worst case percentage saving of components shown in the table are as follows; 37.5% for the adder, -25% for the shifter, 37.5% for the register, 25% for the intermediate memory buffer and 0% for the re-process memory buffer. As the increase in area (due to the extra shifter) is small, the impact to the saving in term of area and power of the overall system will not be affected.

6. CONCLUSION

A low hardware count parallel shift-accumulator centric architecture has been presented which will lead to a low power lifting-based 2D-DWT architecture. We have shown that this architecture can lead to significant reduction in area, which will eventually result in lower power consumption DWT hardware architecture. The above reduction is achieved by reducing hardware requirement of the DWT processors through careful design and scheduling.

Acknowledgement

This work is supported by the Engineering and Physical Sciences Research Council (EPSRC) of United Kingdom under grant number GR/N08322.

7. 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. Website: <http://www.jpeg.org/CDs15444.html>.
- [5] K. Andra, C. Chakrabarti and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform", *IEEE Transactions on Signal Processing*, Vol. 50, No.4, pp.966-977, Apr 2002.
- [6] C-J. Lian, K-F. Chen, H-H.Chen and L-G. Chen, "Analysis and architecture design of lifting based DWT and EBCOT for JPEG2000", *Proceedings of Technical Papers of 2001 International Symposium on VLSI Technology, Systems and Applications*, pp.180-183, 2001
- [7] W-H. Chang, Y-S. Lee, W-S. Peng and C-Y. Lee, "A line-based, memory efficient and programmable architecture for 2D DWT using lifting scheme", *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems*, ISCAS 2001, Vol. 4, pp.330-333, 2001.
- [8] K. C. B. Tan and T. Arslan, "Low power embedded extension algorithm for the lifting-based discrete wavelet transform in JPEG2000", *IEE Electronics Letters*, Vol. 37, No.22, pp.1328-1330, Oct 2001.
- [9] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression", *IEEE Transactions on Image Processing*, Vol. 9, No. 3, pp.378-389, Mar 2000.
- [10] 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.
- [11] 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.