EFFICIENT HARDWARE IMPLEMENTATION OF DISCRETE WAVELET TRANSFORM

Awais M. Kamboh

Electrical and Computer Engineering, Michigan State University, East Lansing.

ABSTRACT

This paper improves upon the lifting scheme and the B-spline implementation to achieve computationally efficient hardware implementation of Discrete Wavelet Transform (DWT). This novel technique involves factoring the wavelet filters into two portions; the first factor contains integer multiplications exclusively while the second factor contains floating point multiplications as well. The efficiency comes from the fact that integer multiplications can be represented by 'shift and add' operations. Examples show that our implementation outclasses both the lifting and B-spline optimizations. Calculation of Shannon entropy and log energy for basis selection is also addressed evaluating their suitability of implementation in hardware.

1. INTRODUCTION

Wavelet packet decomposition (WPD) allows the freedom of choice to the designer for the selection of desired basis owing to the over-complete dictionary of orthogonal basis produced by the WPD. Discrete Wavelet transform, on the other hand, does not provide any over-complete set of basis to choose from, however, it is still used more often then WPD especially in scenarios where hardware implementation of algorithms is required as in portable devices. In this paper we look at these algorithms from a hardware engineer's point of view where area and power are the two main resource constraints. Higher number of computations map to more area and higher power dissipation. One of the most convincing reasons for the under-use of WPD is the computational complexity associated with its implementation. Second reason is the cumbersome hardware required for implementation of selection criteria for basis, i.e. the Shannon entropy and log energy. Both of these quantities depend upon calculation of logarithm, which is a highly resource consuming calculation step. Logarithms cannot be calculated precisely, approximation of log functions also produces reasonable results and the amount of precision required for these approximations is one of the implementation decisions which significantly alter the total amount of required calculations. Higher precision requires higher resources and larger word length.

The paper discusses the two major types of calculations reduction techniques i.e. the popular lifting scheme and a relatively new B-spline scheme in section 2, section 3 gives the details of our hybrid technique that reduces the number of calculations. Section 4 compares the results from the three techniques presented. Section 5 discusses the implementation of logarithmic function in hardware followed by conclusions and future vectors in last section.

2. COMPUTATION REDUCTION TECHNIQUES

2.1. Lifting implementation

Use of lifting steps to reduce the complexity of filters for wavelet transform is a well known and popular technique. The filters are broken down into simpler filters that 'lift' each one of them in terms of the other. The lifting of low pass filter of DWT with the help of high pass filter is called the primal lifting step, the reverse is called dual lifting step. *S(z)* and *T(z)* in (1) are essentially these primal and dual lifting filters. *P(z)* in (1) is called the Polyphase matrix.

$$P(z) = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} \prod_{i=1}^{n} \left\{ \begin{pmatrix} 1 & S_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ T_i(z) & 1 \end{pmatrix} \right\} \quad (1)$$

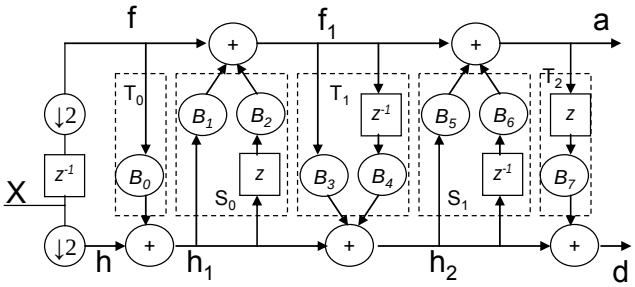

Fig. 1: Lifting Architecture

Fig. 1 is the hardware implementation of (1). The lifting steps have an effect which can be explained more appropriately as predict and update steps. The resulting filters require almost half the number of multiplications and additions as compared to the conventional convolution based filtering. The last step of multiplication with K_1 and K_2 has not been shown.

2.2. B-spline implementation

B-spline has recently emerged as another factorization technique that could be helpful in reducing the number of total computations. This technique is particularly useful for hardware implementations. This technique exploits the fact that integer multiplications can be represented by a collection of 'shift and add' operations. In binary arithmetic, multiplication of a number by 2^n can be implemented by shifting the binary representation left by one bit. So 3 represented in binary as 011 can be shifted left one bit resulting in 110 which is the binary representation of 6. Similarly divide operations can be performed by shifting the bits to the right. The importance of this operation in the filtering operation can be show with the help of following example.

$$\tilde{H}(z) = -\frac{1}{8}z^{-2} + \frac{1}{4}z^{-1} + \frac{3}{4} + \frac{1}{4}z - \frac{1}{8}z^{2} \quad (2)$$

$$\tilde{L}(z) = \frac{1}{4}z^{-2} - \frac{1}{2}z^{-1} + \frac{1}{4}$$

$$\tilde{H}(z) = (z^{-2} + 2z^{-1} + 6 + 2z + z^{2})(-\frac{1}{8}) \quad (3)$$

$$\tilde{L}(z) = (z^{-2} - 2z^{-1} + 1)(\frac{1}{4})$$

The two filters given in (2) have 8 multiplications and 6 additions collectively, per sample, however if we rewrite the equations in form of (3), then it can be seen that all the multiplications can be performed by shifting left and the final divisions can be performed by shifting right. In this paper the additions and subtractions are all referred to as additions as both operations require almost the same hardware. Another thing to note is that the shifting operations do not require any substantial extra hardware and do not introduce any additional delay. In this paper the shift operations will be considered cost-free. As a result of replacing the multiplications with additions and shifts we can implement these two filters with 7 additions in all per sample. This is a huge reduction in complexity as compared to the 8 multiplications and 6 additions per sample of convolution based filtering.

B-spline exploits this property of integer multiplications. The DWT filters are factorized in the form of equations (4).

$$H(z) = (1+z)^{M} Q(z) \cdot h_o \quad (4)$$
$$L(z) = (1-z)^{M} R(z) \cdot l_o$$

where

$$(1 \pm z^{-1})^4 = (1 + 6z^{-2} + z^{-4}) \pm (4z^{-1} + 4z^{-3}) \quad (5)$$

$(1 \pm z^{-1})^M$ will be called the B-spline part, the filters $Q(z)$ and $R(z)$ are called the distributed parts and h_o and l_o are

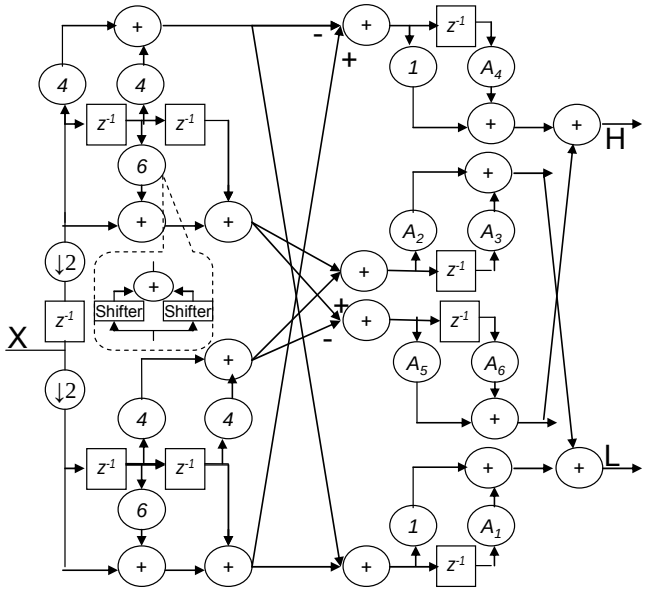

Fig. 2: B-spline Architecture

constants. The expansion of B-spline part in (5) reveals that it requires only integer multiplications. This portion can be implemented using shifts and adds. The distributed portion, however, requires floating point multiplications. Fig. 2 shows the implementation of filters using B-spline where the left half of diagram comprise of shifters and adders only, while the right half of the diagram requires floating point operations as well which is essentially the implementation of distributed filters $Q(z)$ and $R(z)$.

3. EFFICIENT REPRESENTATION

In our implementation we have combined both lifting and B-spline into one to get an efficient algorithm for calculating filter results. The idea is to factorize the filters into $(1 \pm z^{-1})^M$, $Q(z)$ and $R(z)$. If $H(z)$ and $L(z)$ are of the order of N and we factor $(1 \pm z^{-1})^M$ then $Q(z)$ and $R(z)$ are of the order of $(N-M)$. Then lifting factorization is applied to $Q(z)$ and $R(z)$, which by virtue of their lower order require smaller number of primal and dual lifting steps, which in turn translates to lower number of calculations.

$$P(z) = \begin{pmatrix} (1+z)^M K_1 & 0 \\ 0 & (1-z)^M K_2 \end{pmatrix} \prod_{i=1}^{n} \left\{ \begin{pmatrix} 1 & S_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ T_i(z) & 1 \end{pmatrix} \right\} \quad (6)$$

To represent the system in terms of polyphase matrix, (1) gets modified as (6) where $S(z)$ and $T(z)$ are different from the corresponding filters in (1). The constant gains K_1 and K_2 are multiplied with $(1 \pm z^{-1})^M$ at the output which can still be computed with only one multiplication.

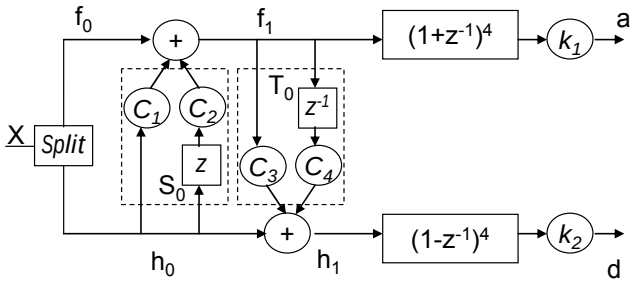

Fig. 3: Efficient Architecture

Fig. 3 gives the graphical representation of (6) where the split of input stream into odd and even samples is followed by lifting steps and the final filtering with the integer part if followed by the floating point multiplication with gains.

This representation collects the notable qualities of both the computation reduction techniques into one technique and is highly suitable for hardware implementation of these filers. It can also be viewed as 'extraction' of integer multiplications from the lifting filters in order to be able to replace multiplications with shifts and adds.

4. COMPARISON

This section compares the three discussed in terms of number of calculations required to compute results for one input sample. We use a common example to compare the results.

(7)

$$H(z) = -0.076 - 0.030z^{-1} + 0.498z^{-2} + 0.804z^{-3} + 0.298z^{-4} - 0.099z^{-5} - 0.013z^{-6} + 0.032z^{-7}$$

$$L(z) = -0.032 - 0.013z^{-1} + 0.099z^{-2} + 0.298z^{-3} - 0.804z^{-4} + 0.498z^{-5} + 0.030z^{-6} - 0.076z^{-7}$$

We choose 'Symlet 4' as a standard for comparison in our example. The low and high pass filters are given in (7). Symlet family of wavelets has gained much popularity in study of neural data. It has been shown to provide high compression ratios on neural signals for data compression and transmission.

TABLE 1: COMPARISON OF REQUIRED COMPUTATION

	Convolution Based	Lifting	B-spline	*Lifting + Bspline
Multiplications	14	10	14(8)	8
Additions	14	8	16(18)	14

* Estimated

Table 1 compares the number of computations required for these schemes. If convolution based conventional filtering is performed then it is evident from (7) that the system required 14 multiplications and 14 additions per sample to generate results. If we perform lifting decomposition, then the number of computations gets reduced significantly to 10 multiplications and 8 additions. When B-spline is performed on the same filters, the number of computation become 14 multiplications and 16 additions, which is even more than the convolution based filtering, but since six of these multiplications can be replaced with shifts and adds, the end results comes down to 8 multiplications and 18 additions.

TABLE 2: COMPARISON COMPUTATION IN TERMS OF ADDERS

Coeff.	B_0	B_1	B_2	B_3
Value	0.3911	-0.1243	-0.3392	-1.4195
Coeff.	B_4	B_5	B_6	B_7
Value	0.162	0.4312	0.1459	-1.0492

TABLE 3: COEFFICIENTS OF THE B-SPLINE FACTORIZATION

H(z)			L(z)		
A_0	A_1	A_2	A_3	A_4	A_5
-3.607	1.867	-0.425	4.391	8.485	2.339

Figures 1 and 2 are true representations of the hardware requirements for these filters. The coefficient values for these decompositions are presented in table 2 and 3 respectively. When we apply our efficient algorithm to these filters, we can perform the same calculations with 8 multiplications and 14 additions. However, we regard these figures as estimated because we do not have exact coefficient values to present. The calculation of coefficient values is, nevertheless, secondary in importance and does not alter the effectiveness of this technique. Note that the Fig. 3 is not a true representation as the number of multiplications and additions essentially depends upon the update and predict filters.

TABLE 4: COMPARISON COMPUTATION IN TERMS OF ADDERS

	Convolution Based	Lifting	B-spline	*Lifting + B-spline
Total Additions	126	88	82	78

*Estimated

There is no specific rule that can compare the cost of implementation of a multiplier with an adder. They change significantly with the changing requirements of area, power or speed of operation. Based on empirical values reported in literature we can estimate an 8x8 multiplier to be more or less equivalent to 8 adders in cost. Using this equality we can compare the computations of these algorithms to each other. The results are presented in table 4. We can clearly see that our algorithm outclasses both of these schemes and would perform even better as the length of filters increases.

This reduction to complexity, however, has some cost associated with it. The property to note here is that B-spline representation is always causal. Lifting on the other hand, can be non-causal, depending on the individual filters, which would require future samples for the calculation of current values, in other words, non-causality would introduce latency of a few samples in the system and it would not remain real-time. Since our efficient algorithm captures the complexity reduction techniques of both lifting and B-spline, the down side is that it looses its causality.

5. LOGRITHMIC FUNCTION

Calculation of logarithm is important for calculation of Shannon entropy (8) and log energy (9) which are in turn important for basis selection algorithms.

$$E = -\sum_i p_i \log(p_i) \tag{9}$$

$$E = \sum_i \log(p_i^2) \tag{10}$$

Exact calculation of logarithm is computationally very intense. The authors are not aware of any existing techniques for exact calculation of logarithms. There are a few techniques available that calculate logarithm in analog domain using non-linearities of semi-conductor devices. This however cannot be done in digital domain in which we are interested. One way to approximate logarithms is by the use and truncation of Taylor series. Another way to approximate is by the use of look up table. Look up table technique is used in state-of-the art microprocessors and digital signal processors for calculation of sine and cosine functions. The same can also be employed for logarithmic functions. Abed and Siferd [3] have discussed the implementation of binary logarithm in detail. Another technique is to calculate $\log_2$ as discussed in [3] and then convert it to $\log_{10}$ based on (11). This conversion does not require any resources other than an adder which adds a constant.

$$\log_{10}X = \log_2X + \log_{10}2 \tag{11}$$

Here we reproduce the summary of ideas presented in [3]. Any number N can be represented as (12)

$$N = \sum_{i=j}^{k} 2^i z_i = 2^k (1 + \sum_{i=j}^{k-1} 2^{i-k} z_i) \tag{12}$$

where z_i is either 0 or 1. Let

$$m = \sum_{i=j}^{k-1} 2^{i-k} z_i \tag{13}$$

then

$$N = 2^k(1+m)$$
$$Log_2N = k + log_2(1+m) \tag{14}$$

m is the binary fraction or the mantissa. Abed and Siferd approximate $\log_2N$ by $(\log_2N)'$ and $\log_2(1+m)$ by a straight line each two successive powers-of-two points. They get the approximation as in (15)

$$(log_2N)' = k + log_2(1 + m) \tag{15}$$

The error is equal to

$$Error = log_2N - (log_2N)'$$
$$= log_2(1+m) - m$$

6. CONCLUSIONS AND FUTURE WORK

We have presented a novel idea of combining the properties of two computation reduction techniques into one technique and our results show that this technique is computationally less complex than either of the two. We have also reproduced the basis for design of an algorithm to calculate logarithmic functions for basis selection in wavelet packet transform.

11. REFERENCES

[1] Karim G. Oweiss, “A Systems Approach for Data Compression and Latency Reduction in Cortically Controlled Brain Machine Interfaces.” IEEE Transactions on Biomedical Engineering, 2006

[2] I. Daubechies and W. Sweldens, “Factoring wavelet transforms into lifting steps,” J. Fourier Anal. Appl.4(3), (1998), pp. 245-267

[3] K. H. Abed, R. E. Siferd, “CMOS VLSI Implementation of 16-bit Logarithm and Anti-logarithm converters”, 43rd IEEE Midwest Symp on Circuits and Systems, 2000.

[4] C. Valens, “The Fast Lifting Wavelet Transform” http://perso.wanadoo.fr/polyvalens/clemens/transforms/wavelet/index.html

[5] C.T. Huang, P.C. Tseung, L.G. Chen, “VLSI Architecture for Forward Deiscrete Wavelet Transform Based on B-spline Factorization”, Journal of VLSI processing, 2005.