HIGH-SPEED LOW COMPLEXITY QUADRATURE MIRROR FILTERS FOR WAVELET-BASED DNA MICROARRAY IMAGE PROCESSING

Zhu Xiangfeng, A.P.Vinod, E.M-K.Lai and Douglas Maskell
School of Computer Engineering, Nanyang Technological University
Nanyang Avenue, Singapore 639798
Email: asvinod@ntu.edu.sg

Abstract: The computational cost of quadrature mirror filters (QMF) used in wavelet-based DNA microarray image processing systems is dominated by the complexity of the coefficient multipliers. Even though many algorithms have been proposed for image denoising using wavelet transform, little attention has been paid to the hardware-efficient realization of wavelet filter banks. In this paper, we present a realization technique for low complexity high-speed QMF for DNA microarray image processing applications. Our algorithm exploits the fact that when multiplication is realized using shifts and adds together, the adder width can be minimized by limiting the shifts of the operands to shorter lengths. Design examples show that the proposed method offers an average full adder reduction of 40% over conventional filter bank implementation methods.

1. Introduction

Microarray is a cutting edge technology in genomic signal analysis that allows thousands of specific DNA sequences to be detected simultaneously on a small glass slide, and permit all of this information to appear on a single image [1, 2]. The basic goal of microarray image processing is to transform an image of spots of varying intensities into a matrix, called a gene expression matrix, which is a measure of the intensity of each spot. Although this transformation is a relatively straightforward goal, the presence of detector noise such as photon noise, electronic noise, laser light reflection and background fluorescence make it a complex process. Noise reduction is a predominant issue in microarray analysis [3]. Wavelet-based denoising algorithms have great potential in DNA microarray image denoising [4, 5]. Even though many algorithms have been proposed for image denoising using wavelet transform, little attention has been paid to the hardware-efficient realization of wavelet filter banks. In order to process the huge amount of data generated by the DNA microarray, dedicated hardware optimized wavelet filter banks are necessary. Enhancing the processing speed of the wavelet filter banks is crucial to achieve real-time image denoising.

In this paper, we present a design technique for realizing low complexity high-speed wavelet filter banks for DNA microarray image denoising. It is well known that tree-structured quadrature mirror filter (QMF) banks are employed in the implementation of discrete wavelet transform [6]. Basically, a QMF bank decomposes the input signal into low-pass and high-pass frequency bands. When filtering is recursively applied to the low-pass frequency bands, the QMF filter bank produces an octave band split or wavelet decomposition. The core operation in a QMF tree for DNA microarray image denoising is two-dimensional (2-D) finite impulse response (FIR) filtering, which requires a huge amount of computations and memory. Hence, the computational complexity must be reduced so as to meet the real-time requirements of microarray image processing. In our method, we implement the 2-D QMF banks as a delayed-sum of 1-D multiplierless QMF filters. The number of computations is reduced by sharing the common terms (common subexpressions) between the coefficients in each 1-D filter.

The paper is organized as follows. A review of filter coefficient multiplier complexity analysis and common subexpression elimination (CSE) [7] method are provided in section 2. In section 3, we present coefficient-partitioning (CP) [8] method. In section 4, we provide design examples to illustrate our method and in Section 5 we provide our conclusions.

2. Multiplier Complexity

For completeness, a brief review of the complexity of the multiplier block (MB) implementation in terms of full adders (FAs) required for each multiplier of the filter formulated in [8] is presented here. Further, we present the CP algorithm and show that the FA requirement can be reduced considerably using our method.

An adder that adds two n-bit numbers requires at most (n+1) FAs to compute the sum. The area, power, and speed of an adder depend on the adder width (n+1). Therefore, the number of full adders (NFAs) required to implement the multipliers must be minimized. Filter coefficients in canonical signed digits (CSD) formed with wordlengths of up to 24 bits are considered for analyzing the adder complexity. Since no adjacent bits in CSD are one's, a 24-bit CSD number can have a maximum of 12 nonzero operands could occur in multiplication.

Case I: Odd number of operands: The NFAs (N_o), required computing the output corresponding to a coefficient with n operands can be determined using the expression [8]:

$$N_o = (r_2+1) + a_1(r_3+1) + (2r_4+3) + a_3(r_5+1) + (r_6+1) + a_5(2r_7+3)$$
$$+(3r_8+6) + a_7(r_9+1) + (r_{10}+1) + a_9(2r_{11}+3) \qquad (1)$$

where r_n is the range (number of bits) of the *n*th operand and a_is are equal to 0 except an-2, which is 1.

Case II: Even number of operands: The NFAs (N_e) required to compute the output corresponding to a coefficient with n operands is given by [8]:

$$N_e = (r_2+1) + (2r_4+3) + c_0(r_6+c'_0) + (3r_8+6) +$$
$$c_1(r_{10}+c'_1) + (3r_{12}+6) \qquad (2)$$

Where $c_0 \equiv \begin{cases} 2, \text{for } n=6 \\ 1, \text{elsewhere} \end{cases}$, $c'_0 \equiv \begin{cases} 1.5, \text{for } n=6 \\ 1, \text{elsewhere} \end{cases}$,

$c_1 \equiv \begin{cases} 2, \text{for } n=10 \\ 1, \text{elsewhere} \end{cases}$, and $c'_1 \equiv \begin{cases} 1.5, \text{for } n=10 \\ 1, \text{elsewhere} \end{cases}$.

The coefficient $h_k = 0.0101001010000101$, is used as an example to illustrate the CSE method [7] here. In direct implementation, (i.e., the implementation using shifts and adds and without CSE or any other multiplier optimization techniques) the output of the filter could be expressed as:

$$y_k = 2^{-2}x_1 + 2^{-4}x_1 + 2^{-7}x_1 + 2^{-9}x_1 + 2^{-14}x_1 + 2^{-16}x_1 \qquad (3)$$

where x_1 is the input. In this case, n is 6 (even), r_2, r_4, and r_6 are 12, 17 and 24 respectively. Using (2) the total number of FA required to compute (3) in direct method is $(r_2+1)+(2r_4+3)+2(r_6+1.5)$, , i.e. 106 FAs.

The goal of the CSE [7] method is to identify multiple occurrences of identical bit patterns in the coefficient set. The pattern [1 0 1] is present thrice in this example, which can be expressed as a common subexpression (CS),

$$x_2 = x_1 + x_1 >> 2 \qquad (4)$$

where ">>" represents 'shift right' operation.

Using the CS (4), the output can be expressed as

$$y_k = x_2 >> 2 + x_2 >> 7 + x_2 >> 14 \qquad (5)$$

Figure 1 shows the multiplication structure using CSE. The numerals adjacent to the data path represents the number of bitwise right shifts. The numerals in brackets alongside the adders indicate the number of FAs used in the adder. In this case, 53 FAs are required for computing y_k using CSE method [7], which is a reduction of 50% over the direct implementation.

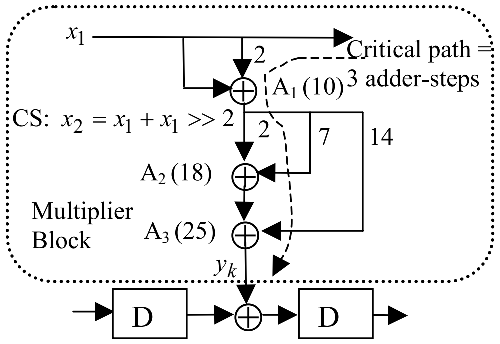

Figure 1: FIR multiplier realization using CSE [7].

3. Coefficient-Partitioning

The key idea in our approach is to reduce the *ranges* of the operands so that the adder width can be reduced which in turn minimizes the number of FAs. To achieve this, firstly the coefficients are encoded using the PFP representation and then partitioned for further reduction of range.

Definition1 (Pseudo floating-point (PFP) representation): The general representation of CSD for the i^th filter coefficient that has a wordlength B is $h_i = \sum_{j=0}^{B-1} 2^{a_{ij}}$. The PFP representation of h_i is [9]

$$h_i = 2^{a_{i0}} . \sum_{j=0}^{B-1} 2^{a_{ij}-a_{i0}} = 2^{a_{i0}} . \left[\sum_{j=0}^{B-1} 2^{c_{ij}} \right] \qquad (6)$$

where $c_{ij} = a_{ij} - a_{i0}$. The term a_{i0} is known as the *shift* and the upper limit value, $(a_{i(B-1)} - a_{i0})$, is known as the *span.* Instead of expressing the coefficients using *B*-bit CSD, it can be expressed as a (*shift, span*) pair using fewer bits. For example, the PFP form of the coefficient in the example in Figure 1 is $2^{-2}(2^0 + 2^{-2} + 2^{-5} + 2^{-7} + 2^{-12} + 2^{-14})$. The term 2^{-2} is the *shift* part, and the bracketed term is the *span* part. The shift operation can be performed after the addition of all the terms of the span part. This reduces the effective wordlength of the coefficient to that of the span (11 bits), which in turn reduces the ranges of the operands. Using (2), the number of FAs required to implement the PFP coefficient multiplier is 98. We shall now show that by combining the PFP coding scheme with the CSE and then partitioning the resulting expression, further reduction of FAs can be achieved.

3.1 FA Reduction Using Coefficient-Partitioning

The basic idea of CP is to reduce the range of the span part of PFP by partitioning it into two parts.

Definition 2 (Order): The most significant bit of a filter coefficient represented in CSD form is defined as the order of the coefficient.

Firstly, the CSD coefficient is expressed using CS and the resulting expression is then coded using PFP representation. Let M represents the span of the PFP representation. The span part is partitioned into two parts of length $M/2$ (or two sub-components of lengths $\lfloor M/2 \rfloor$ and $\lceil M/2 \rceil$ if M is odd). The latter sub-component is then scaled by its *order* to reduce its span. The 'partitioned and scaled' versions of the PFP coefficients thus obtained can be added using fewer numbers of FAs since their ranges are reduced. Consider the same example of the filter tap shown in Figure 1. Using PFP, the filter output obtained in CSE method (5) can be expressed as $2^{-2}(x_2 + 2^{-5}x_2 + 2^{-12}x_2)$. In this case, the span (M) is

9 and the shift is 5. Partitioning the span part into two parts, $h_1(n)$ and $h_2(n)$, we have

$$h_1(n) = x_2 \text{ and } h_2(n) = 2^{-5}x_2 + 2^{-12}x_2 \qquad (7)$$

where $h(n)$ is the sum of $h_1(n)$ (MSB half) and $h_2(n)$ (LSB half). The LSB part is further scaled by its order, 2^{-5}, and expressed as $h_2(n) = 2^{-5}(x_2 + 2^{-7}x_2)$. Figure 2 shows the implementation of the filter tap using our CP method. When compared with the CSE method in Figure 1, the adders A_2 and A_3, have shorter widths since the ranges of their operands are shorter. The shift 2^{-5} of $h_2(n)$ and that of the final expression $2^{-2}(x_2 + 2^{-5}x_2 + 2^{-12}x_2)$ are performed after the addition stages as shown alongside the data paths at the outputs of adders A_2 and A_3 respectively.

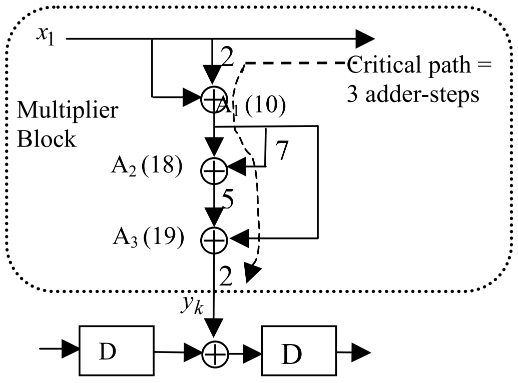

Figure 2: Multiplier realization using CP method.

Thus, our CP method requires only 47 FAs to implement the filter tap, which is a reduction of 11.3% compared with the CSE method [7]. Note that both methods have identical critical path lengths (3 adder-steps) and hence their multiplier delays are same.

The steps of the CP algorithm are as follows.

Step 1: Design the filter of length *N*.
Step 2: Obtain the CSD representation of the coefficients for a desired wordlength. Set $k = 0$.
Step 3: Identify the CS [1 0 1] and [1 0 –1] and their negated versions in $h(k)$. Express the filter output corresponding to the coefficient $h(k)$ using HCSE.
Step 4: Express the HCSE output corresponding to $h(k)$ in PFP. Set $M = span$.
Step 5: Partition the span part into two parts of length $M/2$. Scale the latter part by its *order*.
Step 6: Increment *k*. If $k \neq N$, go to Step 3. Otherwise, terminate the program.

4. Design Examples

Example 1: A prototype QMF that has pass-band and stop-band edges 0.5π and 0.52π respectively is considered. The filter specific taps vary from 20 to 400 and the coefficient wordlength using CSD is from 8-bit to 24-bit. The reduction of adders achieved using the CSE method over the direct implementation is shown in Figure 3. The average reduction of adders using CSE method [7] over direct method for N=50, 80, 120 and 250 are 28%, 28%, 27% and 24% respectively.

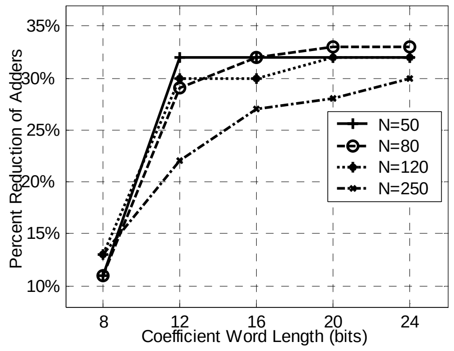

Figure 3: Adder reduction using CSE method over direct method

Figure 4 shows that when the coefficient wordlength is 16-bit, the adder reductions for N=50, 80, 120 and 250 are 32%, 32%, 30% and 27% respectively. Average reduction of adders using CSE method [7] over direct method is 31%.

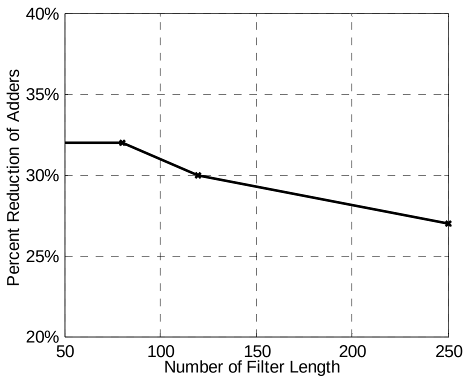

Figure 4: Adder reduction for 16-bit coefficient wordlength

Figure 5 shows that when the coefficient wordlength is 24-bit, the adder reductions for N=50, 80, 120 and 250 are 32%, 33%, 32% and 30% respectively. Average reduction of adders using CSE method [7] over direct method is 32%.

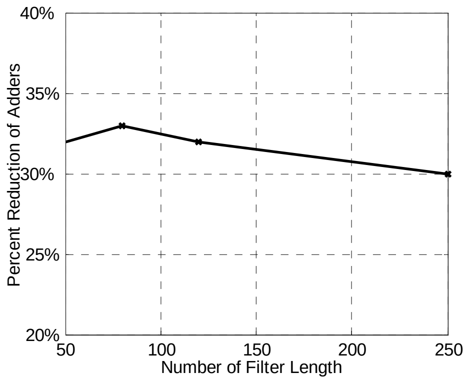

Figure 5: Adder reduction for 24-bit coefficient wordlength

The percentage reduction of FAs using CSE method combined with our CP method over direct implementation is shown in Figure 6. The average reduction of FAs using CP-CSE method over direct implementation for N=50, 80, 120 and 250 are 44%, 44%, 43% and 42% respectively.

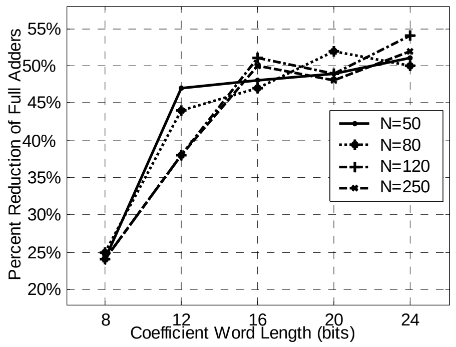

Figure 6: FA reduction using CP-CSE method over direct method.

Figure 7 shows that when the coefficient wordlength is 16-bit, the adder reductions for N=50, 80, 120 and 250 are 48%, 41%, 51% and 50% respectively. Average reduction of adders using CP-CSE method over direct method is 48%.

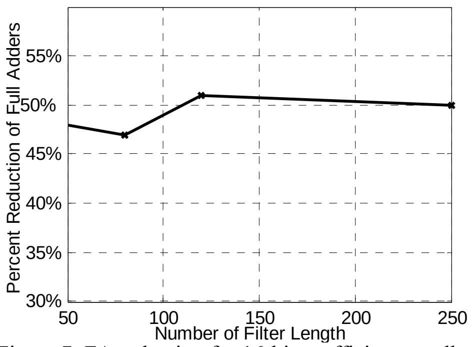

Figure 7: FA reduction for 16-bit coefficient wordlength

Figure 8 shows that when the coefficient wordlength is 24-bit, the reduction of the adder for N=50, 80, 120 and 250 are 51%, 50%, 54% and 52% respectively. Average reduction of adders using CP-CSE method over direct method is 51%.

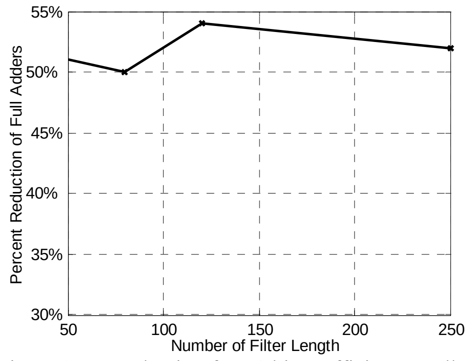

Figure 8: FA reduction for 24-bit coefficient wordlength

5. Conclusions

We have proposed a design technique for realizing low complexity high-speed quadrature mirror filter for DNA microarray image denoising. The design examples show that our method offers average FA reductions of 40% over direct implementation. Though the design examples are shown for a prototype quadrature mirror filter, our method can be easily extended for a tree-structured QMF bank, which is the fundamental building block of wavelet filter banks.

6. References

[1] Y. H. Yang, M. J. Buckley, S. Dudiot, and T. P. Speed, "Comparison of methods for image analysis on cDNA microarray data," Journal of Computational and Graphical Statistics, vol. 11, pp. 108-136, January 2002.

[2] P. P. Vaidyanathan, "Genomic and proteomics: A signal processor's tour," IEEE Circuits and Systems Magazine, vol. 4, pp. 6-29, Fourth Quarter 2004.

[3] X-Y. Zhang, F. Chen, Y-T. Zhang, S. C. Agner, M. Akay, Z-H. Lu, M. M. Y. Waye, and S. K-W. Tsui, "Signal processing techniques in genomic engineering," Proceedings of the IEEE, vol. 90, no. 12, pp. 1822-1832, December 2002.

[4] X. H. Wang, R. S. H. Istepanian, and Y. H. Song, "Microarray image enhancement by denoising using stationary wavelet transform," IEEE Transactions on NanoBioscience, vol. 2, no. 4, pp. 184-189, December 2003.

[5] F. E. Turkheimer, D. C. Duke, L. B. Moran, and M. B. Graeber, "Wavelet analysis of gene expression (WAGE)," Proceedings of IEEE International Symposium on Biomedical Imaging: Macro to Nano, vol. 2, pp. 1183-1186, April 2004.

[6] P. P. Vaidyanathan, *Multirate Systems and Filter Banks:* Prentice Hall, 1993.

[7] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," IEEE Trans. Circuits Syst. II, vol. 43, pp. 677-688, Oct. 1996.

[8] A. P. Vinod and E. M-K. Lai, "An efficient coefficient-partitioning algorithm for realizing low complexity digital filters," *IEEE Trans. On Computer-Aided Design of Integrated Circuits Syst.,* vol. 24, no. 12, Dec. 2005 (To appear).

[9] A. P. Vinod, A. B. Premkumar and E. M-K. Lai, "An optimal entropy coding scheme for efficient implementation of pulse shaping FIR filters in digital receivers," *Proceedings of the IEEE International Symposium on Ckts. And Syst.,* vol. 4, pp. 229-232, Bangkok, Thailand, May 2003.