The Design of Multiplierless FIR Filters with a Minimum Adder Step and Reduced Hardware Complexity

Douglas L. Maskell, Jussipekka Leiwo and Jagdish C. Patra
School of Computer Engineering
Nanyang Technological University
Singapore

***Abstract*—We propose an algorithm for reducing the hardware complexity of linear phase FIR digital filters without resorting to an increase in the number of adder steps in the multiplier block adders. We aggressively reduce both the coefficient wordlength and the number of non-zero bits in the filter coefficients so that the adder step can be minimized. The hardware implementation of the coefficients is such that the number of full adders is proportional to the product of the input signal wordlength and the number of adders. That is, in general, the number of full adders is independent of the coefficient wordlength and the number of shifts between non-zero bits in the coefficient. Results show that the proposed technique achieves a 67% and 71% reduction in the number of multiplier block adders and the number of multiplier block full adders respectively. Our technique has been successfully applied to filters with up to 500 taps.**

I. INTRODUCTION

Finite impulse response (FIR) digital filters have found widespread use in digital signal processing and communications applications because of their stability, linear phase and simple regular structure. In many applications, these filters are required to be implemented in hardware rather than as software algorithms. In today's consumer driven embedded systems domain, constraints such as low power consumption and very high data throughput mean that general purpose programmable hardware filters are unsuitable. Instead, the filter's constant coefficient multipliers are usually implemented using a sequence of shift and add operations. These implementations are referred to as multiplierless structures and present the filter designer with a number of conflicting design issues. The complexity of the design problem becomes evident when one considers the large design space and the need for optimizing several incompatible and often competing objectives. These include area, power, and throughput as well as the traditional objectives of the coefficient precision, the number of additions, the adder depth, etc.

Previous research into efficient multiplierless filter implementations has been conducted in two independent sub-areas, namely: discrete optimization of the quantized filter coefficients [1]-[8]; and the reduction in the number of multiplier block adders [9]-[13]. Many of these coefficient optimization techniques [4] are very compute intensive and are applicable to filters with only a small number of taps. As a result, other sub-optimal techniques such as random local search [5], tree search [2], simulated annealing [7] and genetic algorithm [8] methods have been proposed. On the other hand, multiplier block reduction has attempted to minimize various cost functions such as adder depth, the number of adders and more recently multiplier block area. Most multiplier block minimization techniques can be categorized as either graphical [9] or subexpression elimination [10]-[13]. Common subexpression (CSE) techniques attempt to minimize the number of additions in the multiplier block by reusing terms. These terms can be canonic signed digit (CSD) [10], [11], minimal signed digit (MSD) [12], or all signed digit (ASD) [13].

While there has been some attempt to combine both research areas [14], implementations that achieve a minimal hardware area satisfying a given frequency specification [15], [16], are only suitable for filters with a very small number of taps. In this paper, we present an integrated suboptimal design technique which firstly attempts to aggressively reduce the coefficient wordsize and the number of non-zero bits in the filter coefficients while maintaining the original filter frequency specifications. Secondly, the minimum worst-case adder depth using horizontal CSE (HCSE) sharing is determined. Constrained by the minimum adder depth, the depth of the adder structure is relaxed to give a significantly reduced FA count and hence a reduced hardware complexity. Thirdly, the filter is generated as synthesizable Verilog HDL code and is suitable for filters with a much larger number of taps than existing techniques [15], [16].

II. PROBLEM DEFINITION

Much of the current research into multiplier block minimization techniques has concentrated in reducing the number of word-wide adders, and has not concentrated on the actual hardware resources needed. Recently, the number of FAs to implement a multiplication by a filter coefficient has been addressed in [11]. The concept of adder span is introduced, such that if a subexpression is formed, say x_2, where $x_2 = x + x>>2$ corresponding to [101], then x_2 has a span of W+2, where W is the wordlength of the input signal (x). Consider the following example [11]:

Let $h_j = 0.100101010101$ and x in this example is quantized to 8-bits. The output can be expressed as:

$$y_j = x>>1 + x_2>>4 + x_2>>8 + x>>12$$

where $x_2 = x + x>>2$ and we are only considering subexpressions [101] and [1̲01].

The coefficient multiplier can be implemented using a linear adder structure with an adder depth of 4[1] or as a binary tree structure with an adder depth of 3. The number of FAs is determined by the spans of the operands as

$$\begin{aligned} N_{linear} &= 14 + 18 + 20 = 52 FAs \\ N_{tree} &= 14 + 2*20 = 54 FAs \end{aligned} \quad (1)$$

The number of FAs needed to compute a result [11] is underestimated because the possibility of an overflow is not considered. That is the product x2 = x + x>>2 corresponding to x*[101] has a span W+2+1, not W+2 as reported in [11].

We propose a different method for assigning FAs which significantly reduces the hardware count compared to [11], while placing a constraint on the maximum adder depth that is allowed. To illustrate this technique, consider a filter coefficient hk = 0.10010101 (0.58203125), where D_{min} has previously been calculated as having a depth of 3 (based on a different coefficient). Note that the product by h_k could be implemented using a linear adder structure with a resulting adder depth of 3, or with a tree structure resulting in an adder depth of 2. As D_{min} is already fixed as 3, both of these are satisfactory and is illustrated below. Here we choose x as an 8-bit value (W = 8) such that x = 0.9296875 = 0.1110111. The values in bold represent a necessary (hard wired) sign extension, while the underlined section represents the actual addition process, and hence the number of FAs.

Linear:

```
     000 01110111
   + 0 01110111↓↓
     00 01001010011
   + 0 01110111↓↓↓↓
     000 0100111000011
   + 0 01110111↓↓↓↓↓↓↓
     0100010101000011
```

Tree:

```
     0000 01110111      000 01110111
   + 0 01110111↓↓↓  + 0 01110111↓↓
     010000101111     01001010011

          00000 01001010011
        + 010000101111↓↓↓↓
          0100010101000011
```

[1] The adder depth is the critical pathin the implementation of a filter coefficient. A coefficient can have different adder depths depending on the implementation structure. The smallest adder depth able to implement all of a particular filter's coefficients is called the minimum adder depth (D_{min}).

Both solutions use the same number of adders (3) and produce the same result ($x*h_k$ = 0.100010101000011 = 0.541107178). However, they use a different number of FAs. The linear implementation requires 3(W+1) = 27 FAs, while the tree implementation requires 30 FAs. The implementation as in [11] requires 38 FAs (40 if we include the sign extensions necessary to cater for any overflow). Our technique for implementing the filter coefficient represents a 32% saving in adder area compared to [11].

This very simple example using only positive numbers shows that it is possible to greatly reduce the number of FAs needed to implement a constant coefficient multiplier. When we consider the other common subexpression, [1̲01], we can again produce a result using *W*+1 FAs. For instance, *x**[1̲01] where *x* = 0.9296875 = 0.1110111, gives a result ($x*h_k$ = 1.1010011011 = -0.348632813) as shown below (LHS). However, a problem arises when we try to generate *x**[101̲], or any summation which has a negative sign associated with the RHS summand. For example, *x**[101̲] where *x* = 0.9296875 = 0.1110111, gives a result ($x*h_k$ = 0.0101100101 = 0.348632813) requiring W+1+2 FAs (see RHS below).

```
     000 01110111          111 10001000
               1                     1
   + 1 10001000↓↓        + 0 0111011100
     11010011011           00101100101
```

To overcome this, we carry to the left the (negative) sign of any addition involving a negative RHS summand as shown in the simple example below. This could eventually result in a negative sign being carried all the way up to the structural adder associated with that coefficient. Modifying the filter coefficients in this way ensures that the maximum number of FAs for a linear implementation of a constant coefficient is (*l*-1)*(*W*+1), that is the number of FAs is independent of the coefficient wordsize and the shift amount between non-zero digits.

Original bit pattern	Modified bit pattern
0.100101	0.100(101)
0.1̲00101	0.1̲00(101)
0.100101	0.100(1̲01)
0.100101̲	-0.1̲00(1̲01)
0.1̲00101	0.1̲00(101)
0.100101̲	-0.100(1̲01)

III. PROPOSED METHOD

The problem to be solved can now be described as: *Given a frequency/deviation specification, generate a filter implementation which minimizes the latency (the adder depth) and the hardware area (The number of FAs).*

One method of reducing the adder depth is to implement each coefficient (h_j) as a separate binary tree of adders. Other approaches [9], [12] result in a smaller number of adders but at the expense of adder depth. Pipeline stages can be added, negating some of the latency issues relating to an increased adder depth. However, the addition of pipeline latches would result in an increase in the filter area, negating to some extent the reduction in the number of adders. We examine techniques for reducing the adder depth and hence the filter latency and filter area.

A. *Coefficient Optimization*

Because discrete coefficient optimization techniques based on mixed integer linear programming are extremely compute intensive, we instead attempt to find a much faster local solution which satisfies the original frequency specifications, while aggressively minimizing both the number of nonzero bits and the coefficient wordlength. To achieve this we use a modification of the Hooke and Jeeves pattern search algorithm implemented in MATLAB. This modified algorithm looks for a solution based upon an initial coefficient word size and an initial number of nonzero CSD bits. If a solution is not found, it incrementally increases the number of nonzero CSD bits followed by an increase in the coefficient wordlength until an acceptable solution is found. Aggressively minimizing the number of nonzero bits results in a significant reduction in the adder depth as adder depth is directly related to the number of bits in the coefficient.

B. *Horizontal Common Subexpression Elimination*

Common subexpression elimination [10], based on the four most common 2-bit subexpressions, is performed on the filter coefficients. These subexpressions are [101], [1̲01], [1001] and [1̲001]. We use these subexpressions rather than the two most common subexpressions as in [10] or higher order combinations, such as 3-bit or 4-bit horizontal super-subexpressions [11]. This is because the routing channel in modern devices (ASIC or FPGA) has increased significantly since the observation [10] was made that the use of more than the two most common subexpressions will lead to increased signal routing costs. Super-subexpressions are not considered as our aggressive minimization of the nonzero digits results in few 3-bit or 4-bit common subexpressions occurring in the filter coefficients making it uneconomical to route these signals through the multiplier block.

C. *Increasing the Adder Depth up to D_{min}*

The minimum adder depth (D_{min}) is calculated. It should be noted that only a small fraction of the coefficients are constrained by D_{min}. We now take the remaining bits in the coefficients after common subexpression elimination and implement them (where possible) as a linear array of adders with a depth up to D_{min}. As shown in a previous section, a linear array of adders requires a smaller number of FAs. The resulting structure is then automatically generated as synthesizable Verilog HDL code and targeted to a Xilinx Spartan 3s1500fg320-5 using Xilinx's Project Navigator.

IV. EXPERIMENTAL RESULTS

To allow space for the optimizer, the number of filter taps is increased by a small amount. This approach is commonly used in filter coefficient optimization [6], [14]. A simple experiment, based on a lowpass filter specification, was conducted to examine the number of extra taps that are appropriate in this situation. The filter in this case had a passband ripple of 0.1dB, a stopband attenuation of -50dB with passband and stopband cutoff frequencies of 0.1π and 0.15π respectively. A number of filters were designed using MATLAB's Remez function. The minimum tap filter which satisfies this specification has 103 taps. Filters with a 3% (106), 5% (108), 8% (111) and 10% (113) increase in the number of taps were designed. These filters were analyzed using our proposed algorithm and the number of MB FAs, structural FAs, and the total number of FAs were compared for each implementation. These results are shown in Table 1. An input signal (x) wordlength of 16 bits is assumed.

TABLE I
NUMBER OF FULL ADDERS TO IMPLEMENT THE FILTER

# Taps	% inc.	MB FAs	SB FAs	Total
106	3%	642	3045	3687
108	5%	566	2996	3562
111	8%	535	3080	3615
113	10%	452	3136	3588

Table 1 shows that the filter with the 5% increase in the number of taps gives the fewest FAs. The filter with the 3% increase in the number of taps has more MB FAs and SB FAs after optimization than for the filter with 108 taps as the wordlength and number of non-zero bits needed to meet the specifications is larger. Based on these figures, a 5% increase is used in the following experiments. Our proposed algorithm was applied to several FIR filters whose specifications are summarized in Table 2. Here, f_p is the normalised passband frequency, f_s is the normalised stopband frequency, A_p is the passband ripple, A_s is the stopband attenuation and *# Taps* is the number of filter taps. The figure in brackets represents the minimum number of taps to satisfy the infinite precision filter specifications.

The filter coefficients were quantized using two techniques for comparison purposes. Firstly, the coefficients were rounding to the smallest bit precision which still satisfies the original frequency specifications, and secondly, the coefficients were determined using our proposed algorithm. This second process involves optimization of the CSD filter coefficients, HCSE, and adder depth modification for coefficients less than the minimum adder depth (D_{min}). These results are presented in Table 3. The rounded coefficient filters correspond to the rows indicated by an * in the wordlength (*WLen*) column, while the remaining rows represent filters designed using our proposed algorithm. The values in the HCSE column represent the MB adders using the 4 most common subexpressions, with the 2 most common subexpression values in brackets. In the multiplier

block full adders (MB FAs) column, the rows indicated by an * represent the number of FAs using the conventional method with the adder span method [11] in brackets. For our proposed method, the values represent the 4 most common subexpressions, with the 2 most common subexpression values in brackets, and are derived directly from the Xilinx synthesis report. The input signal (x) wordlength is 16 bits.

TABLE II
TEST FILTER SPECIFICATIONS

Filter	f_p	f_s	A_p	A_s	# *Taps*
1	0.1	0.15	0.1	-50dB	108 (103)
2	0.2	0.225	0.05dB	-50dB	222 (212)
3	0.2	0.2125	0.05dB	-50dB	441 (420)

TABLE III
MB ADDERS AND FULL ADDERS FOR EXAMPLE FILTERS

		MB Adders			
Filter	*WLen.*	*# Steps*	*Simple*	*HCSE*	*MB FAs*
1	13*	3	91	44 (54)	1740 (1024)
	12	2	67	33 (41)	586 (731)
2	15*	3	208	98 (130)	4303 (2698)
	14	3	165	73 (98)	1295 (1713)
3	17*	3	521	248 (303)	11632 (6799)
	16	3	367	148 (201)	2683 (3431)

From Table 3 we see that there is a significant reduction in both the number of MB adders and the number of FAs using our proposed algorithm. There is an average 67% reduction in the number of MB adders for our algorithm when compared to the simple conventional implementation. This can be compared to the 25% to 44% reduction in the number of MB adders presented in [11] and the average 61% reduction (with an increase in the number of adder steps) presented in [12]. It should be noted that there are a number of algorithms which are able to reduce the number of adders at the expense of an increase in the number of adder steps [12], [17]. However, one of our initial design criteria was that the minimum adder step not be exceeded.

In terms of the number of FAs, there is a 71% reduction in the number of FAs for our algorithm compared to a simple conventional implementation. For implementations using the two most common subexpressions there is an average 63% reduction in the number of FAs. These results can be compared to the average 40% saving using the adder span method [11] as calculated from Table 3, and the 25% to 54% reduction in the number of FAs as presented in [11].

V. CONCLUSIONS

In this paper we have proposed an algorithm for reducing the hardware complexity of linear phase FIR digital filters without resorting to an increase in the number of adder steps in the multiplier block adders. We aggressively reduce the number of non-zero bits in the filter coefficients so that the adder step can be reduced. We also propose a modification to the representation of the filter coefficients such that the number of full adders in our hardware implementation is proportional to only the product of the signal wordlength and the number of adders. Results show that there is a 67% and 71% reduction in the number of MB adders and the number of MB FAs respectively. These results are significantly better than other results presented in the literature.

REFERENCES

[1] F. Ashrafzadeh, B. Nowrouzian and A.T.G. Fuller, "A novel modified branch-and-bound technique for discrete optimization over canonical signed-digit number space", *IEEE ISCAS*, vol. 5, pp. 391-394, 1998.

[2] Y.C. Lim and S.R. Parker. "FIR filter design over a discrete powers-of-two coefficient space." *IEEE Trans. Acoust., Speech, Signal Processing*, vol ASSP-31, pp. 583-591, June, 1983.

[3] C.-L. Chen and A.N. Willson Jr.. "A trellis search algorithm for the design of FIR filters with signed powers-of-two coefficients." *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 29-39, Jan. 1999.

[4] Y. C. Lim. '"Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude", *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1480-1486, Dec. 1990.

[5] Q. Zhao and Y. Tadokoro. "A simple design of FIR filters with powers-of- two coefficients", *IEEE Trans. Circuits Syst.*, vol. 35, pp. 566-570, May. 1988.

[6] H. Samueli. "An improved search algorithm for the design of multiplierless FIR filters with power-of-two coefficients", *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044-1047, July 1989.

[7] J. Radecki, J. Konrad and E. Dubois, "Design of Multidimensional Finite-Wordlength FIR and IIR Filters by Simulated Annealing", *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 424-431, Jun. 1995.

[8] D.J. Xu and M.L. Daley, 'Design of optimal digital filter using a parallel genetic algorithm", *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 673 - 675, Oct. 1995.

[9] A.G. Dempster and M.D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters", *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 569-577, Sep. 1995.

[10] R.I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers", *IEEE Trans. Circuits Syst. II*, vol. 43, Oct. 1996.

[11] A.P. Vinod and M-K Lai, "On the implementation of efficient channel filters for wideband receivers by optimizing common subexpression elimination methods", *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 24, pp. 295-304, Feb. 2005.

[12] I-C Park and H-J Kang, "Digital filter synthesis based on an algorithm to generate all minimum signed digit representations", *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 21, pp. 1525-1529, Dec. 2002.

[13] A.G. Dempster and M.D. Macleod, "Generation of signed-digit representations for integer multiplication", *IEEE Signal proc. Lett.*, vol. 11, pp. 663-665, Aug. 2004.

[14] M. Bhattacharya and T. Saramäki, "Some observations on multiplierless implementation of linear phase FIR filters", *Proc. 2003 IEEE ISCAS*, vol 4, pp. 193-196, May 2003.

[15] K-H Tan, W.F Leong, K. Kaluri, M.A. Soderstrand and L.G. Johnson, "FIR filter design program that matches specifications rather than filter coefficients results in large savings in FPGA resources", *Asilomar Conf. Sig., Sys. & Comp.*, vol 2, pp. 1349-1352, Nov. 2001.

[16] J. Yli-Kaakinen and T. Saramaki, " A systemic algorithm for the design of multiplierless FIR filters", *Proc. 2001 IEEE ISCAS*, vol. 2, pp 185-188, 2001.

[17] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde and D. Iuraekova, "A new algorithm for elimination of common subexpressions", *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 18, pp. 58-68, Jan. 1999.