Hardware-efficient FIR filters with reduced adder step

D.L. Maskell and J. Liewo

A technique for reducing the hardware complexity of constant coefficient finite impulse response (FIR) digital filters, without increasing the number of adder steps in the multiplier block adders, is presented. The filter coefficients are adjusted so that the number of full adders in the hardware implementation of any coefficient is independent of the coefficient wordlength and the number of shifts between nonzero bits in the coefficient. Results show that the proposed technique achieves a significant reduction in both the multiplier block adders and the multiplier block full adders when compared to existing techniques.

Introduction: Finite impulse response (FIR) digital filters are widely used in DSP applications because of their stability, linear phase response and their simple regular structure. Constraints such as area, speed and power mean that the filter's constant coefficient multipliers are usually implemented in hardware using a sequence of shift and add operations. These structures 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 optimising several incompatible and often competing objectives.

Much of the current research into efficient multiplierless filter implementations has been conducted in two independent sub-areas, namely: discrete optimisation of the quantised filter coefficients and the reduction in the number of multiplier block adders. While there has been some attempt to combine both research areas, these techniques are only suitable for filters with a very small number of taps [1]. Similarly, while there has been considerable research effort expended in reducing the system complexity at the adder level, there is little analysis of the actual hardware complexity of these adders [2]. In this Letter, we present a technique for significantly reducing the FA count and hence the hardware complexity, constrained by the minimum adder depth of the filter.

Structural implementation: A constant coefficient multiplier can have different adder depths depending on the implementation structure [3]. The adder depth influences the critical path delay, as well as the area and power. The smallest adder depth able to implement all of a particular filter's coefficients is called the minimum adder depth ($D_{\min}$), and is given by $D_{\min} = \max\{\lceil \log_2 l_{\max} \rceil\}$, where $l_{\max}$ is the maximum number of nonzero bits in any of the filter coefficients.

While most research in this area has concentrated on reducing the number of adders, little analysis of the actual hardware complexity of these adders has been undertaken. Adder complexity is determined by a combination of the coefficient shift amount and the input wordsize [2]. We propose a method which significantly reduces the hardware complexity by removing the dependency on the shift amount. To illustrate this technique, consider the product of x by the subexpressions [1̲01] and [101̲], respectively. For instance x^*[1̲01] where $x = 0.9296875 = 0.1110111$, gives a result ($x^*h_k = 0.1010011011 = -0.348632813$) as shown in Fig. 1*a*. This results in an adder complexity proportional to W, where W is the wordlength of the input signal (x). 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 ($x^*h_k = 0.0101100101 = 0.348632813$) as shown in Fig. 1*b*. This implementation has a complexity proportional to $W+S$, where S is the shift amount; i.e. the complexity is proportional to the input wordsize plus the shift amount. To overcome this limitation, we convert any negative RHS summand to positive and carry the (negative) sign to the left. This could eventually result in a negative sign being moved all the way up to the structural adder associated with that coefficient, as shown in Table 1 (lines 4 and 6). This can easily be extended to a larger coefficient wordsize, and a larger number of nonzero bits. It should be noted here, that subexpressions with identical nonzero bits need an additional bit in the summation to cater for any possible overflow. Performing this modification to a linear implementation of the filter coefficients ensures that the adder complexity is proportional to only the input wordsize and the number of nonzero bits in the coefficient, and has no dependency on either the coefficient wordsize or the shift amount between nonzero digits. Other implementations, such as a tree implementation, still maintain some partial proportionality to the shift amount between nonzero digits.

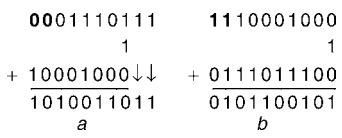

Fig. 1 *Product of 8-bit value (01110111) by CSE [1̲01] requiring W (8) adders (Fig. 1a) and CSE [101̲] requiring W+S (10) adders (Fig. 1b)*
Bold characters represent a necessary hardwired sign extension

Proposed method: In this Section, we outline our methodology for reducing the adder depth and hence the filter latency and filter area. We first apply discrete coefficient optimisation techniques to find a fast local solution which satisfies the original frequency specifications, while aggressively minimising both the number of nonzero bits and the coefficient wordlength. This ensures a significant reduction in the adder depth as adder depth is directly related to the number of bits in the coefficient. Common subexpression elimination (CSE) [4] is then applied. The four most common 2-bit CSEs are chosen as a 2-bit CSE which does not impact on the adder depth. Where possible, adders are implemented as a linear array with a depth up to $D_{\min}$. This is because a linear array has the least hardware complexity. A software program has been implemented which generates a technology independent Verilog HDL description of the digital filter.

Table 1: Rearranging constant coefficient to eliminate adder dependence on shift amount (any resultant negative sign carried forward to structural adder)

Original bit pattern	Modified bit pattern
0.100101	0.100 (101)
0.1̲00101	0.1̲00 (101)
0.100101	0.100 (101)
0.100101	−0.100 (101)
0.100101	0.100 (101)
0.1̲00101̲	−0.100 (101)

Experimental results: First we examined a number of constant coefficient multipliers. Our technology independent technique for implementing the filter coefficient represents an average 32% saving in adder area compared to [2] and an average 44% reduction in FPGA resource compared to a technology dependent commercial constant coefficient multiplier generator [5].

Next, we examined several FIR filters. These results are presented in Table 2. For the rows indicated by an *, the entries correspond to rounded coefficient filters, while the other rows correspond to filters designed using our technique. The values in the HCSE column represent the MB adders using the four most common 2-bit subexpressions. The number of full adders required by the implementation is given in the multiplier block full adders (MB FAs) column. For the rounded coefficient filters these are determined using the conventional method, while for the other rows the number of FAs are determined using our technique. For comparison purposes, the values in brackets represent the number of FAs using the adder span method [2].

Table 2: Number of MB adders and MB full adders for example filters

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

Table 2 shows that using our technique there is a significant reduction in both the number of MB adders and the number of FAs. There is an average 67% reduction in the number of MB adders for our algorithm

when compared to the simple conventional implementations. This can be compared to an average 43% reduction using SLRAGn with a minimum adder depth as presented in [3], and an average 61% reduction (with an increase in the number of adder steps) as presented in [6]. It should be noted that there are a number of algorithms which are able to reduce the number of adders at the expense of adder step [6], however one of our initial design criteria was that the minimum adder step and hence the critical delay path should be constrained. When the number of FAs are compared, there is a 71% reduction in the number of FAs for our algorithm when compared to the simple conventional implementation. Our technique represents an additional 20% saving over the adder span implementation method [2].

Conclusion: We have proposed an algorithm for reducing the hardware complexity of constant coefficient FIR digital filters without resorting to an increase in the number of adder steps in the multiplier block adders. 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, respectively, in the number of MB adders and the number of MB FAs. These results are significantly better than other results presented in the literature.

15 July 2005
Electronics Letters online no: 20052559
doi: 10.1049/el:20052559

D.L. Maskell and J. Liewo (*School of Computer Engineering, Nanyang Technological University, Singapore*)

E-mail: asdouglas@ntu.edu.sg

References

1 Yli-Kaakinen, J., and Saramaki, T.: 'A systemic algorithm for the design of multiplierless FIR filters'. Proc. 2001 IEEE Int. Symp. Circuits and Systems, Sydney, Australia, 2001, Vol. 2, pp. 185–188
2 Vinod, A.P., and Lai, M.-K.: 'On the implementation of efficient channel filters for wideband receivers by optimizing common subexpression elimination methods', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2005, **24**, pp. 295–304
3 Kang, H.-J., and Park, I.-C.: 'FIR filter synthesis algorithms for minimizing the delay and the number of adders', *IEEE Trans. Circuits Syst. II*, 2001, **48**, pp. 770–777
4 Hartley, R.I.: 'Subexpression sharing in filters using canonic signed digit multipliers', *IEEE Trans. Circuits Syst. II*, 1996, **43**, pp. 677–688
5 Xilinx Corporation, 'Multiplier Generator v7.0', April 28, 2005
6 Park, I.-C., and Kang, H.-J.: 'Digital filter synthesis based on an algorithm to generate all minimum signed digit representations', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2002, **21**, pp. 1525–1529