Redundancy Reduction for High-Speed FIR Filter Architectures Based on Carry-Save Adder Trees

Anton Blad and Oscar Gustafsson
Electronics Systems, Linköping University,
SE-581 83 Linköping, Sweden
E-mail: antonb@isy.liu.se, oscarg@isy.liu.se

***Abstract*—In this work we consider high-speed FIR filter architectures implemented using, possibly pipelined, carry-save adder trees for accumulating the partial products. In particular we focus on the mapping between partial products and full adders and propose a technique to reduce the number of carry-save adders based on the inherent redundancy of the partial products. The redundancy reduction is performed on the bit-level to also work for short wordlength data such as those obtained from sigma-delta modulators.**

I. Introduction

The two main architectures for FIR filters are the direct-form (DF) architecture and the transposed direct-form (TF) architecture. In this paper, both architectures are implemented using partial product generation to realize the filter coefficient multiplications, and carry-save adders (CSA) to efficiently reduce the number of partial products. Finally, the CSA output is merged to a non-redundant binary output using a vector merge adder (VMA). For generality we will primarily consider polyphase decomposed FIR filters. These have applications when the sampling rate is higher than the clock frequency and can also be straightforwardly simplified to decimation or interpolation filters, as well as traditional single-rate filters. As high-speed filters are considered, the partial product reduction trees are strongly pipelined, and the focus in this paper is on the implementation of such reduction trees. In addition to high speed, the work focuses on short wordlength filters, i.e., 1–8 bits. Typical applications of such filters are decimation filters for sigma-delta modulators.

The DF architecture and the TF architecture are depicted in Figs. 1 and 2, respectively. Both figures show multirate structures with K input branches and N output branches. In the DF architecture, delay chains at the input provide the algorithmic delays, and adder trees sum the partial products generated from the taps. In the TF architecture, the partial products are generated directly from the inputs, and the delay elements in the reduction tree serve as both pipeline delays and algorithmic delays. For high-speed filters, the TF architecture may therefore provide a more register-efficient realization. However, as the number of cascaded adders in each pipeline stage is restricted, there may be additional CSA stages needed to reduce the number of partial products sufficiently before the VMA.

For both the DF architectures and the TF architecture, the result after partial product generation is a number of partial

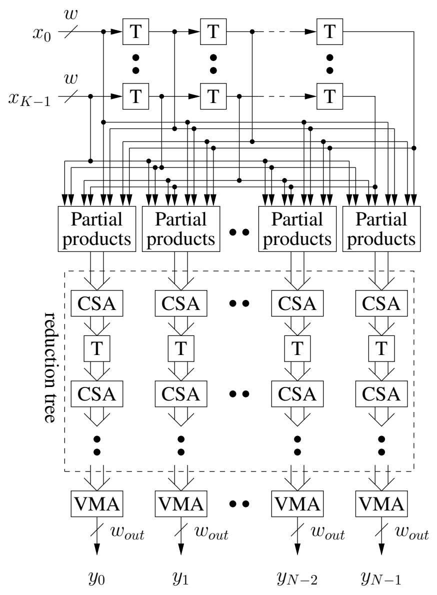

Fig. 1. Multirate direct-form (DF) FIR filter architecture.

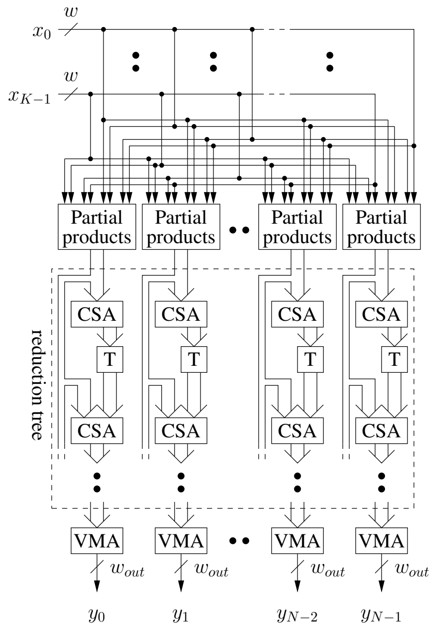

Fig. 2. Multirate transposed direct-form (TF) FIR filter architecture.

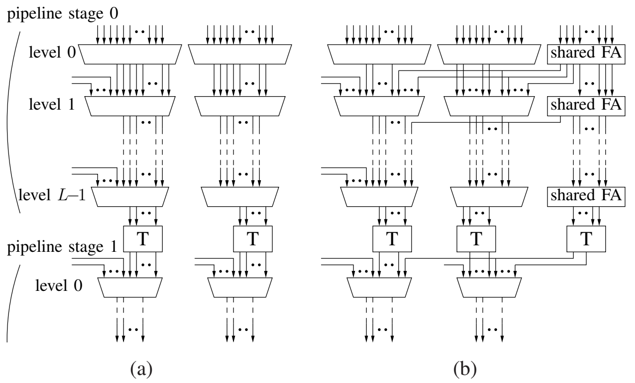

Fig. 3. Pipelined partial product reduction tree with L levels. Each level has a depth of at most one full adder. (a) Original reduction tree. (b) Reduction tree utilizing subexpression sharing.

products with different bit weights and different delays. In [1]–[4], different methods of generating partial products for given filters were investigated. The method discussed in this work can be applied to any type of partial product generation scheme. However, the redundancy relates to the fact that the same partial products are present in more than one column of the summation tree. Hence, this method is straightforward to apply using the methods in [2]–[4], whereas the method in [1] requires careful investigation to identify identical partial products at the outputs of the different distributed arithmetic blocks.

Methods for constructing reduction trees based on full and half adders include both an optimal integer linear programming (ILP) based method [4] and several heuristics, e.g., Wallace trees [5], Dadda trees [6], delay optimized trees [7], HPM [8], and Reduced Area [9] trees. The ILP based method constructs a reduction tree which is optimal given area costs on full and half adders and registers. The major drawback of this method is that it can become time consuming to solve the ILP problem for large trees. The aggressive use of half adders in Wallace trees leads to fast reductions, but generally a more efficient use of half adders is possible. The Dadda structure uses half adders more restrictively only to try to maximize the opportunities to use full adders. However, only placing full adders as late as possible makes the structure unsuitable for pipelining. The Reduced area heuristic is taking advantage of the major benefits of the Wallace and Dadda tree, providing early reduction using full adders and a sensitive use of half adders. This is beneficial from a pipelining point of view as it results in fewer registers compared to Dadda trees, while still keeping the number of half adders small. The delay optimized approach takes advantage of the different delays of the carry and sum outputs of a full adder and produces a tree with small delay. With pipelining this benefit is typically reduced. Finally, the HPM tree provides regular layout for reduction trees originating from multipliers, while still achieving a logarithmic reduction delay. For other types of reduction trees it is in general not possible to obtain regular layouts for logarithmic reduction.

It was claimed in [9] that the Reduced Area heuristic is optimal in terms of hardware resources for general multipliers, but simulations provided in [4] show that this is not necessarily true for general partial product trees. Moreover, none of the heuristics consider that partial products might be added with different delays in the reduction tree for certain types of FIR filter architectures, e.g., the TF architecture or other examples covered in [4].

The utilization of computational redundancy in FIR filters has been a well studied problem for the last decade. However, most work is based on carry-propagation adders and primarily transposed direct form FIR filters, see e.g. [10] for an overview of previous work. By utilizing the redundancy between the coefficients, it is possible to realize several constant multiplications sharing the same input using a small number of adders and subtracters[1]. However, carry-propagation adders are not well suited for high-speed operations, even though there are methods to reduce the number of cascaded adders, the adder depth. Instead, there has been some methods for redundancy reduction using carry-save adders [11].

The problems with the previous approaches are here briefly discussed. While transposing the network of carry-propagation adders results in the same complexity, one does not have any control of the resulting adder depth. The carry-save adder approaches have not explicitly considered the adder depth, but instead relied on the fact that it is at least faster than using carry-propagation adders. Furthermore, it is not possible to consistently transpose carry-save adder structures. This means that to obtain low adder depth, one should compute the redundancy reduction considering the structure to be implemented. In addition to this, for short wordlengths these methods also become unreliable as the relatively large difference in wordlengths between different partial results is not considered. To illustrate this, let us consider the multiplication with 9 (computation of the sub-expression 1001). In general $\lceil \log_2 9 \rceil = 4$ additional bits are required to represent the result. Now, if the input wordlength is small (in this case ≤ 4) the output will have more or as many bits as the two separate inputs. Hence, no actual reduction is obtained. Therefore, for short wordlengths the redundancy reduction should be done on a bit-level basis rather than a word-level basis as all work so far has done.

In this work we will, based on a given reduction tree, identify redundancies between the partial products and use these to reduce the total complexity of the FIR filters.

II. Proposed Approach

A. Preliminaries

The general structure of the original reduction trees is shown in Fig. 3(a). The reduction is done in a number of pipeline stages, separated by registers. Each stage is further divided into a number of levels, each with a depth of at most one full adder. Thus the number of levels is the maximum adder

[1]As the complexity of adders and subtracters are about the same they are commonly referred to as adders.

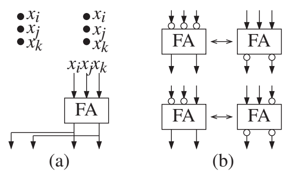

Fig. 4. (a) Illustration of redundancy reduction. (b) Equivalence transform to detect more redundancies.

depth of the reduction tree. Let a subexpression denote three partial products with the same weight, on the same stage and level in the same reduction tree. The sharing is then based on identifying common subexpressions occurring in several places in the reduction trees. Thus the sharing identification is performed before the assignment of partial products to adders. Subexpressions can be shared in the same tree or between trees, and in the same stage or between stages. However, in order not to affect the delay of the filter, subexpressions can not be shared between different levels. For each subexpression to be shared, the occurrences of the partial products are removed from the reduction trees, an adder for the subexpression is introduced, and partial products corresponding to the sum and carry outputs are added on the level below each subexpression occurrence. The redundancy reduction using adder sharing is depicted in Fig. 4 (a), and the resulting reduction tree structure is shown in Fig. 3(b).

Several different reduction trees can share the same full adder. However, there can not be any full adder that have inputs from more than one adder tree. Hence, the search for possible assignments need to be done inside each adder tree only, but to determine which assignment is most beneficial one should consider all adder trees. For the first level of the first stage all partial products are from the partial product generation stage. For latter levels there may still be redundancy as the same partial product result may end up in several columns (see Fig. 4 (a)). Considering the DF and TF architectures, adder sharing can not be expected to consistently yield better results with one of the architectures than the other. In the DF architecture, all partial products are added in the first level, increasing the number of possible sub-expressions that can be considered for sharing. On the other hand, the delay chain at the input also increases the number of different partial products and decreasing the probability that the same partial products occur in several columns.

If signed data and/or coefficients are used, some of the partial products may be inverted. However, it is possible to apply transformations to increase the possible sharing by propagating inversions from the inputs to the outputs as illustrated in Fig. 4 (b).

B. Proposed Algorithm

The goal of the proposed algorithm is to identify subexpressions that occur more than once. It can be described as follows:

1) Represent the filter coefficients in a suitable representation and generate one or more adder trees to be computed.
2) For each column in each level in each reduction tree, find the possible subexpressions. Note that the subexpressions shall be normalized in terms of possible inversion.
3) Determine the maximum frequency subexpression. In case of a tie, pick one at random.
4) If the chosen subexpression has only one occurrence, finish.
5) Assign the subexpression to a full adder, remove the corresponding entries from the adder trees, and add the sum and carry outputs as partial products in the corresponding columns at the level below each subexpression occurrence.
6) Go to 2.

III. Results

In this section, adder sharing is evaluated for a number of filters. Filters for which bit-level optimization is expected to be particularly useful are multi-rate FIR filter implementations of CIC decimators and interpolators. CIC decimators with high data rate and short wordlengths are commonly used for sigma-delta modulators. Thus, as evaluation filters CIC decimation and interpolation filters with sampling rate change factors of four and eight were used. Wordlengths between one and six bits were used for the decimation filters, and wordlengths between three and eight bits were used for the interpolation filters. Both direct form and transposed direct form structures were evaluated, and for all filters the resulting adder trees were bit-level optimized after all shared sub-expressions were identified and adders introduced. Binary arithmetic with a critical path of three adders were used in all cases. The savings in register complexity were limited, and thus only the savings in adder complexity are presented.

Three filters were used in cascade, and thus the single-rate impulse responses for the example filters are

$$h_4(k) = (1,3,6,10,12,12,10,6,3,1)\,, \text{ and} \tag{1}$$

$$\begin{aligned} h_8(k) = (&1,3,6,10,15,21,28,36,42,46,48,\\ &48,46,42,36,28,21,15,10,6,3,1)\,, \end{aligned} \tag{2}$$

for the filters of factors 4 and 8, respectively.

All the filter examples in this section have been generated with the authors' high-speed FIR filter generator. [2]

Results of adder sharing for the decimation filters are shown in Fig. 5. The bigger filter offers significantly more sharing potential, which is expected. Generally, increasing the data wordlength increases the number of partial products and therefore the sharing potential.

Results of adder sharing for interpolation filters are shown in Fig. 6. Since the multiple output branches of interpolation filters cannot be merged into a common adder tree as for the decimation filters, the number of candidate sub-expressions is significantly reduced and the sharing potential is smaller. This is especially true for the transposed direct form, where the

[2]Available at: `http://www.es.isy.liu.se/software/hsfir/`

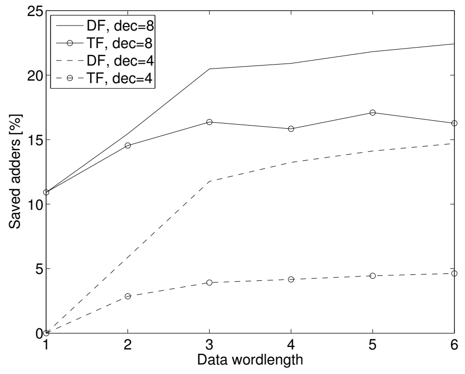

Fig. 5. Multirate FIR CIC decimation filters with factors of 4 and 8.

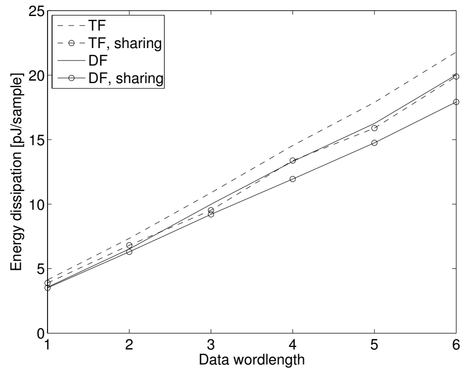

Fig. 7. Energy dissipation estimation of FIR CIC decimation filters with factor of 8.

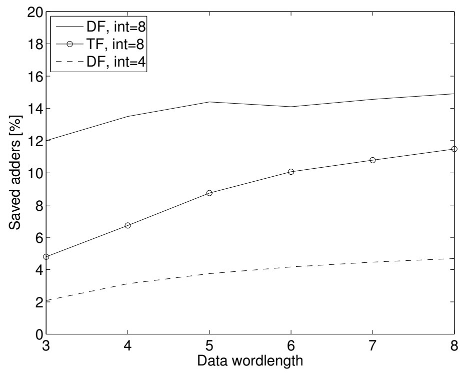

Fig. 6. Multirate FIR CIC interpolation filters with factors of 4 and 8.

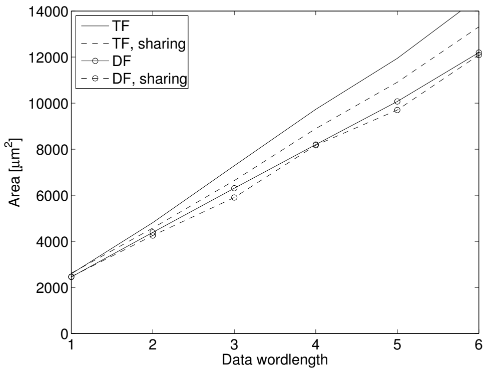

Fig. 8. Area estimation of FIR CIC decimation filters with factor of 8.

structural delays in the reduction tree also inhibits merging of partial products from all taps in each output branch. For the TF architecture with interpolation by four there was no sharing potential.

The factor-8 decimation filters in Figs. 5 and 6 have also been synthesized to standard cells of a 90nm CMOS process using Synopsys Design Compiler. Flipflops, full adders and half adders were used as leaf components, and the designs were synthesized for a clock frequency of 1.4 GHz. Energy and area estimations are shown in Figs. 7 and 8 respectively. It is seen that the redundancy reduction reduces both the energy dissipation and the area requirements.

IV. Conclusions

High-speed FIR filter architectures using partial product generation and carry-save adder trees were considered. A method to detect redundant adders in the reduction tree was proposed, and evaluated using multi-rate FIR filter structures for CIC decimation and interpolation transfer functions. Significant reductions of the number of adders were achieved even for small filters. Furthermore, energy and area estimations based on a 90nm CMOS standard cell library was performed.

References

[1] C. Kuskie, B. Zhang, and R. Schreier, “A decimation filter architecture for GHz delta-sigma modulators,” in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, May 1995, pp. 953–956.

[2] H. Aboushady, Y. Dumonteix, M.-M. Louerat, and H. Mehrez, “Efficient polyphase decomposition of comb decimation filters in $\Sigma\Delta$ analog-to-digital converters,” *IEEE Trans. Circuits Syst. II*, vol. 48, no. 10, pp. 898–903, Oct. 2001.

[3] O. Gustafsson and H. Ohlsson, “A low power decimation filter architecture for high-speed single-bit sigma-delta modulation,” in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, May 2005, pp. 1453–1456.

[4] A. Blad and O. Gustafsson, “Integer linear programming-based bit-level optimization for high-speed FIR decimation filter architectures,” *Circuits Syst. Signal Processing - Special Issue Low Power Digital Filters*, vol. 29, no. 1, pp. 81–101, Feb. 2010.

[5] C. Wallace, “A suggestion for a fast multiplier,” *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[6] L. Dadda, “Some schemes for parallel multipliers,” *Alta Frequenza*, vol. 34, pp. 349–356, May 1965.

[7] V. G. Oklobdzija, D. Villeger, and S. S. Liu, “A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach,” *IEEE Trans. Comp.*, vol. 45, no. 3, pp. 294–306, March 1996.

[8] H. Eriksson, P. Larsson-Edefors, M. Sheeran, M. Själander, D. Johansson, and M. Schölin, “Multiplier reduction tree with logarithmic logic depth and regular connectivity,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, doi: 10.1109/ISCAS.2006.1692508.

[9] K. Bickerstaff, M. Schulte, and E.E. Swartzlander, Jr., “Parallel reduced area multipliers,” *J. VLSI Signal Processing*, vol. 9, no. 3, pp. 181–191, Nov. 1995.

[10] O. Gustafsson, “Lower bounds for constant multiplication problems,” *IEEE Trans. Circuits Syst. II*, vol. 54, no. 11, pp. 974–978, Nov. 2007.

[11] ——, “Comments on ‘A 70 MHz multiplierless FIR Hilbert transformer in 0.35 um standard CMOS library’,” *IEICE Trans. Fundamentals*, vol. E91-A, no. 3, March 2008.