Integer Linear Programming-Based Bit-Level Optimization for High-Speed FIR Decimation Filter Architectures

Anton Blad and Oscar Gustafsson

Anton Blad and Oscar Gustafsson, Integer Linear Programming-Based Bit-Level Optimization for High-Speed FIR Decimation Filter Architectures, 2010, CIRCUITS SYSTEMS AND SIGNAL PROCESSING, (29), 1, 81-101.
http://dx.doi.org/10.1007/s00034-009-9116-5

Title:	Integer Linear Programming-Based Bit-Level Optimization for High-Speed FIR Decimation Filter Architectures
Running head:	ILP-Based Optimization for High-Speed FIR Filters
Authors:	Anton Blad and Oscar Gustafsson
Affiliation:	Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden E-mail: antonb@isy.liu.se, oscarg@isy.liu.se

Abstract

Analog-to-digital converters based on sigma-delta modulation have shown promising performance, with steadily increasing bandwidth. However, associated with the increasing bandwidth is an increasing modulator sampling rate, which becomes costly to decimate in the digital domain. Several architectures exist for the digital decimation filter, and among the more common and efficient are polyphase decomposed FIR filter structures. In this paper, we consider such filters implemented with partial product generation for the multiplications, and carry-save adders to merge the partial products. The focus is on the efficient pipelined reduction of the partial products, which is done using a bit-level optimization algorithm for the tree design. However, the method is not limited only to filter design, but may also be used in other applications where high-speed reduction of partial products is required.

The presentation of the reduction method is carried out through a comparison between the main architectural choices for FIR filters: the direct-form and transposed direct-form structures. For the direct-form structure, usage of symmetry adders for linear-phase filters is investigated, and a new scheme utilizing partial symmetry adders is introduced. The optimization results are complemented with energy dissipation and cell area estimations for a 90nm CMOS process.

Index Terms

FIR, Polyphase, Sigma-Delta, CIC, Optimization, Integer Linear Programming, Decimation, Digital Filter, Carry-Save

I. Introduction

Oversampling can significantly simplify the implementation and/or increase the performance of analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) [7]. As the converters are oversampled the data rate must be changed, and, hence, we must either decimate (ADC) or interpolate (DAC) the signal. In this work we consider decimation filter implementations of high-speed ADCs, specifically those based on $\Sigma\Delta$-modulation [18]. One key feature of $\Sigma\Delta$-modulation, apart from being oversampled, is that the data wordlength is short, often much shorter than the final resolution. Hence, we primarily focus on decimation filters with short wordlengths, but the techniques are generalizable to arbitrary wordlengths. Decimation is often performed in several stages to allow for simpler decimation filters at each stage, so that the overall complexity is reduced [20], [22]. It is worth noting that the wordlength differ between the stages. Especially, for ADCs based on $\Sigma\Delta$-modulation the wordlength often increases significantly after the first stage.

Many decimation filters for $\Sigma\Delta$-modulators are based on cascaded integrator comb (CIC) filters (also known as moving average filters) for the first filter stage. The impulse response of an N-tap (order $N-1$) CIC filter is

$$H(z) = \frac{1}{N}\sum_{i=0}^{N-1} z^{-i} = \frac{1}{N}\frac{1-z^{-N}}{1-z^{-1}} \tag{1}$$

If the number of taps, N, is selected as $N = M$, where M is the decimation rate, the CIC filter has zeros at $\pi i/M$ rad for $i = 1, 2, \ldots, M$, i.e., the angles that are folded to 0 (corresponding to DC) during the decimation. To increase the attenuation, several CIC filters, say L, are cascaded. Such a filter is often referred to as a sinc^L-filter.

One possible realization of a CIC filter is to directly use the rightmost expression in (1) [6], [12]. This leads to a low arithmetic complexity as only $2L$ adders are required. However, the wordlength of the integrators (accumulators) in a CIC filter is significantly longer than the input wordlength, which may lead to problems obtaining high throughput as the integrators operate at the input sampling rate. This can be alleviated by the use of redundant arithmetic in the integrators [16] or parallelizing the integrators to operate at a lower sampling rate.

Recent implementation studies have shown that it is more advantageous, from a power consumption point of view, to compute the impulse response of the cascaded filters and realize the resulting linear-phase FIR filter using polyphase decomposition. Several different architectures have been proposed [1], [10], [11], [15]. Furthermore, using general FIR filters instead of CIC filters allows an arbitrary set of filter coefficients, optimized for a suitable cost function. This is especially useful when using $\Sigma\Delta$-modulators with arbitrary zero positioning for the noise transfer function. In this work we consider the architecture choices for polyphase decomposed FIR filters. The focus is on decimation filters for high-speed $\Sigma\Delta$-modulators, typically in the GHz range [17], so a short critical path is needed in the decimation filter. The examples are based on CIC filters due to their common use in $\Sigma\Delta$-converters, although it is possible to implement arbitrary FIR filters using the same approach.

In [1], [10], [11], [15], different methods of generating partial products for given filters were investigated. However, in this paper the focus is rather on the generation of an efficient pipelined reduction tree. This is done through a formulation as an integer linear programming (ILP) problem, with which a bit-level optimized reduction tree can be obtained. As cost function for the optimization algorithm, a weighted sum of the number of full adders, half adders, and registers is used. The model is similar to that presented in [14], but was not formulated as an ILP problem there.

Compared with the traditional heuristic methods: Wallace trees [21], Dadda trees [8], and Reduced Area [4] trees, the bit-level optimization yields better results for a number of reasons. 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. It is also the case that the heuristics work well for reduction trees for general multipliers but less so for other reduction trees. For example, the Reduced Area heuristic is claimed to be optimal in terms of hardware resources for general multipliers, but simulations provided in this paper show that this is not necessarily the case for general partial product trees. Moreover, the heuristics do not consider that partial products might be added at different levels in the reduction tree, which is the case for several of the architectures considered in this paper.

It should be noted that the techniques in this paper can be applied to reduction trees in other applications as well. One possible application is Merged arithmetic [19], where the results of several multiplications are summed in a carry-save adder tree, similarly to the filter architectures in this paper. Other examples are high-speed complex multipliers implemented using distributed arithmetic [3], and implementation of general functions as sums of weighted bit-products [13].

A preliminary version of this work was earlier presented in [5]. In the current work, a direct-form architecture utilizing partial symmetry adders has been included, different coefficient representations have been investigated, and the optimization problem has been extended to allow signed-digit coefficients. Also, power dissipation estimations have been included.

The rest of the paper is organized as follows. In Sec. II, the considered architectures are described. In Sec. III, theoretical estimates of the architectures' implementation complexities are provided. Formulations of the architecture optimization as ILP problems are presented in Sec. IV, and simulation results are given in Sec. V. Finally, conclusions are in Sec. VI.

II. Architectures

The suitability of the two main architectures for FIR filters are investigated: the direct-form (DF) architecture, and the transposed direct-form (TF) architecture. 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).

The direct-form (DF1) architecture is depicted in Fig. 1 . In this architecture, a delay chain at the input provides the algorithmic delays, and an adder tree sums the partial products generated from the taps. An interesting characteristic of the direct-form architecture is its ability to utilize the symmetry of linear-phase FIR filter coefficients. This architecture is depicted in Fig. 2 , and is denoted the DF2 architecture. The benefit of utilizing the symmetry is that the number of multiplications is halved. However, in the application of high-speed decimation filters, this feature is not necessarily efficient because of the need for short critical paths. The inability to implement the symmetry adders using carry save arithmetic leads to excessive use of registers in pipelined ripple-carry adders. This can be readily observed in the experimental results in Section V. A solution to this problem is suggested in [9], but demonstrated in [11] to have limited efficiency. As a possible solution instead we consider using *partial symmetry*, by this we mean dividing the symmetry adders into smaller blocks of adders as illustrated in Fig. 3 . This will reduce the register complexity, as the carry propagation chain is broken, but leads to slightly more partial products compared to full symmetry. The partial symmetry architecture is referred to as the DF3 architecture.

Fig 1

Fig 2

Fig 3

The TF architecture is depicted in Fig. 4 . For high-speed decimation filters, the TF architecture may provide a more register-efficient realization, as the algorithmic delay elements are also used as pipelining registers in the summation tree. Because of the architecture's limited ability to utilize filter coefficient symmetry, however, the TF architecture may require more adders than the direct-form architecture. The reasons that we cannot utilize symmetry are two: first, the symmetric multiplier may be connected to a different input beacuse of the polyphase decomposition, second, we do not compute each multiplication explicitly, instead we merge all multiplications in a single stage to one carry-save tree. As we restrict the number of cascaded adders in each stage, there may be additional CSA stages to reduce the number of partial products before the VMA. Hence, the output of each CSA stage is not necessarily represented using at most two bits for each bit weight.

Fig 4

For both the DF architectures and the TF architecture, the result after partial product generation is a number of partial products with different bit weights and different delays. The general structure for the summation tree is shown in Fig. 5 , where the carry-save adder is divided into J stages. The stages are separated by pipeline registers, and input is accepted in each stage. Each stage has the structure shown in Fig. 6 , allowing a maximum adder depth of K levels. Again, partial products may be added in every level. Considering the investigated architectures, for the DF1 architecture all partial products are added in the first level of the first stage. For the TF architecture partial products are added in the first level of several stages, as the pipeline registers are also used as algorithmic delays. Finally, partial products are added in several levels of the first stages for the DF2 and DF3 architectures, as the inputs are delayed by the symmetry adders.

Fig 5

Fig 6

A. Partial Product Generation

As the multiplier coefficients are known it is not required to use general multipliers. Instead we generate only the partial products corresponding to non-zero coefficient bits and add these. Here, we will discuss how to generate partial products using different representations of the coefficients and also using both signed and unsigned input data.

An input data X with a wordlength of w_d bits can be written as

$$X = \sum_{i=0}^{w_d-1} x_i 2^i \tag{2}$$

for unsigned data with an input range of $0 \leq X \leq 2^{w_d} - 1$ and

$$X = -x_{w_d-1} 2^{w_d-1} + \sum_{i=0}^{w_d-2} x_i 2^i \tag{3}$$

for signed (two's complement) data with an input range of $-2^{w_d-1} \leq X \leq 2^{w_d-1} - 1$, where for both (2) and (3) $x_i \in \{0, 1\}$. Note that the input data is considered to be integer instead of fractional. However, this is only to be consistent with the numbering used later on, where the bits corresponding to the smallest weight (the LSBs) have index 0.

Similarly, the w_c-bits coefficients, $h(n)$, can be written as

$$h(n) = \sum_{j=0}^{w_c-1} h_{n,j} 2^j \tag{4}$$

and

$$h(n) = -h_{n,w_c-1} 2^{w_c-1} + \sum_{j=0}^{w_c-2} h_{n,j} 2^j, \tag{5}$$

where $h_{n,j} \in \{0, 1\}$.

The output of a $\Sigma\Delta$-modulator is often a positive integer. Therefore, only (2) must be considered. Also, for a cascade of CIC filters all the impulse response coefficients have positive values. Hence, we will initially consider partial product generation of unsigned data and unsigned coefficients. An unsigned multiplication of an input data and a filter coefficient can be written as

$$Xh(n) = \left(\sum_{i=0}^{w_d-1} x_i 2^i \right) \left(\sum_{j=0}^{w_c-1} h_{n,j} 2^j \right) = \sum_{j=0}^{w_c-1} \sum_{i=0}^{w_d-1} h_{n,j} x_i 2^{i+j}. \tag{6}$$

Now, as some of the $h_{n,j}$-bits are known to be zero, we only need to add bits corresponding to non-zero $h_{n,j}$.

If we instead use a signed-digit (SD) representation of the coefficient, i.e., $h_{n,j} \in \{-1, 0, 1\}$, the number of non-zero $h_{n,j}$ can be decreased. A signed-digit representation with a minimum number of non-zero positions is called a minimum signed-digit (MSD) representation. In general there is no unique MSD representation, but introducing the constraint that no two adjacent positions should both be non-zero, i.e., $h_{n,j} h_{n,j+1} = 0$ will result in the canonic signed-digit (CSD) representation, which is both unique and minimum.

Using a SD representation now require that we can subtract partial products instead of just adding them. By enabling subtraction we also at the same time enable signed input data and coefficients. Equation (6) will now contain partial products which may be both positive and negative. Now, note that $-a = \bar{a} - 1$ for $a \in \{0, 1\}$. Hence, negative partial products can be handled by inverting the partial product and adding a constant number. The constant numbers corresponding to all partial products can be merged to one constant binary number in the filter computation. In Fig. 7 the partial products resulting from multiplying a three-bits signed input with the coefficient 29 is illustrated for both binary and CSD representation of the coefficient. The corresponding constants to add are $-4-16-32-64 = -116$ and $-4-4-8-128 = -144$ for the binary and CSD representation, respectively. It should be noted that for the TF architecture, the output will be correct only after the first $\lceil (N-1)/M \rceil$ samples. If correct output from the first sample is required, one solution is custom initialization of each stage register. Fig 7

III. Implementation Complexity

A. Adder complexity

As only the full adders reduce the number of partial products, the required number of full adders for the carry-save summation tree can be easily calculated as the difference between the number of generated partial products and the output wordlength. For the DF1 architecture and the TF architecture, the number of generated partial products can be written $w_d(N+1)N_a$, where N is the filter order, and N_a is the average number of non-zero digits in the filter coefficients. For the DF2 architecture, the number of coefficients is halved, whereas the input wordlength is increased by one due to the symmetry adders. Thus the number of generated partial products can be written $(w_d+1)\left\lceil \frac{N+1}{2} \right\rceil N_a$. Finally, for the DF3 architecture, each symmetry adder increases its input wordlength with one, and hence the total number of partial products can be written $\left(w_d + \left\lceil \frac{w_d}{S} \right\rceil\right)\left\lceil \frac{N+1}{2} \right\rceil N_a$, assuming that partial symmetry adder groups of S bits are used. Depending on the representation used for coefficient and input data there may also be a number of constant ones to add.

In all architectures, the required number of output bits, assuming the general case with signed full-scale data and signed coefficients, can be written $w_{out} = w_d + \log_2 \sum |h(k)|$, where $h(k)$ is the impulse response. Thus the required number of full adders can be written

$$N_{\mathrm{FA,DF1}} = N_{\mathrm{FA,TF}} = w_d((N+1)N_a - 1) - \log_2 \sum |h(k)| \tag{7}$$

for the DF1 architecture and the TF architecture,

$$N_{\mathrm{FA,DF2}} = (w_d+1)\left\lceil \frac{N+1}{2} \right\rceil N_a - w_d - \log_2 \sum |h(k)| \tag{8}$$

for the DF2 architecture, and

$$N_{\mathrm{FA,DF3}} = \left(w_d + \left\lceil \frac{w_d}{S} \right\rceil\right)\left\lceil \frac{N+1}{2} \right\rceil N_a - w_d - \log_2 \sum |h(k)| \tag{9}$$

for the DF3 architecture. Also, the complexity of the symmetry adders for the DF2 architecture is $\left\lceil \frac{N+1}{2} \right\rceil$ w_d-bit adders, resulting in a number of adder cells equal to

$$N_{\mathrm{FA,sym,DF2}} = (w_d - 1) \left\lceil \frac{N+1}{2} \right\rceil \tag{10}$$

and

$$N_{\mathrm{HA,sym,DF2}} = \left\lceil \frac{N+1}{2} \right\rceil. \tag{11}$$

For the DF3 architecture the number of partial symmetry adders is $w_d/S \left\lceil \frac{N+1}{2} \right\rceil$ S-bit adders, resulting in a number of adder cells equal to

$$N_{\mathrm{FA,sym,DF3}} = \left(w_d - \left\lceil \frac{w_d}{S} \right\rceil\right) \left\lceil \frac{N+1}{2} \right\rceil \tag{12}$$

and

$$N_{\mathrm{HA,sym,DF3}} = \left\lceil \frac{w_d}{S} \right\rceil \left\lceil \frac{N+1}{2} \right\rceil. \tag{13}$$

Using these equations, the total required number of adders for the DF architectures can be calculated. It should be noted, however, that the equations (7)–(9) does not take into account the half adders that are usually needed to rearrange the partial products, but nevertheless an approximate condition can be determined for when the DF2 architecture results in a structure with smaller adder complexity compared with DF1. This condition is

$$N_{\mathrm{FA,DF1}} > N_{\mathrm{FA,DF2}} + N_{\mathrm{FA,sym,DF2}} + N_{\mathrm{HA,sym,DF2}}, \tag{14}$$

which can be approximated to

$$w_d > \frac{N_a}{N_a - 1}, \tag{15}$$

if the costs of half and full adders are considered equal. However, as the half adders of the CSA trees have been ignored, the condition should be considered as a guideline rather than as a strict rule. In the investigated application where, typically, both w_d and N_a are low, utilizing the coefficient symmetry often does not lead to reduced adder complexity.

B. Register complexity

Regarding the register complexity, it is possible to find expressions that are asymptotically valid. However, for the considered applications these expressions convey little information, and expressions that are valid for low filter orders and short wordlengths are difficult to find. Thus, the register complexities due to algorithmic delays are calculated here, whereas those due to pipelining of the adder trees are determined experimentally.

For the DF architectures, the algorithmic delays are applied at the input, and the register complexity due to these can be written

$$N_{\mathrm{R,DF1}} = w_d N. \tag{16}$$

If the symmetry of the coefficients is utilized, the implementation carries an additional complexity due to pipelining of the symmetry adders. The way the pipelining is done is shown in Fig. 8 . In addition to the

Fig 8

registers needed to restrict the length of the critical path, registers are also placed at the outputs of the full adders just before the cuts. The reason for this is the definition of the reduction trees in Sec. IV, which does not accept inputs just before pipeline cuts. If an n-bit ripple-carry adder with a maximum of m adders in the critical path is considered, the required number of pipeline registers can be written

$$N_{\mathrm{R,RC}}(n,m) = \sum_{k=1}^{\lfloor n/m \rfloor} 2(n-mk+1). \tag{17}$$

The register complexity of the DF2 architecture can then be written

$$N_{\mathrm{R,DF2}} = w_d N + \left\lceil \frac{N+1}{2} \right\rceil N_{\mathrm{R,RC}}(w_d, K), \tag{18}$$

where K is the maximum allowed number of adders in cascade. For the partial symmetry case, the number of registers is smaller (for $S < w_d$). The number of registers, assuming $S = nK$ for an integer n, is

$$N_{\mathrm{R,DF3}} = w_d N + \left\lceil \frac{N+1}{2} \right\rceil \left(\left\lfloor \frac{w_d}{S} \right\rfloor N_{\mathrm{R,RC}}(S,K) + N_{\mathrm{R,RC}}(w_d \bmod S, K) \right). \tag{19}$$

For (18) and (19) it is assumed that no sharing of registers between the algorithmic delays and symmetry adder pipeline registers has been performed. If sharing is considered, the register complexity of the DF2 architecture can be written

$$N_{\mathrm{R,DF2}} = w_d N + \left\lfloor \frac{w_d}{K} \right\rfloor (N+1) + \sum_{m=0}^{M-1} \sum_{i=0}^{\lceil \frac{N+1-m}{M} \rceil - 1} \sum_{k=1+i}^{\lfloor \frac{w_d}{K} \rfloor} (w_d - Kk). \tag{20}$$

If, for simplicity, $w_d = jS$ for an integer j is assumed, the register complexity of the DF3 architecture can be written

$$N_{\mathrm{R,DF3}} = w_d N + jn(N+1) + j \sum_{m=0}^{M-1} \sum_{i=0}^{\lceil \frac{N+1-m}{M} \rceil - 1} \sum_{k=1+i}^{n} (S - Kk). \tag{21}$$

For the TF architecture, the algorithmic delays are merged with the pipeline registers, and all registers are in the adder tree.

IV. ILP Optimization

Denote the stage height, i.e., the maximum number of cascaded adders, by K, as in Fig. 6. Denote also the number of stages by J, as in Fig. 5. Furthermore, denote the output wordlength by w_{out}, and the number of input partial products in each stage j, level k and bit position b by $INBITS_j(k,b)$, $k \in \{0,1,2,\ldots,K-1\}$, $b \in \{0,1,2,\ldots,w_{out}-1\}$. As variables for the ILP problem, the number of full adders $FA_j(k,b)$ and half adders $HA_j(k,b)$ are used, with the same parameter bounds as $INBITS$. The resulting number of partial products in each level is denoted $BITS_j(k,b)$, and is defined $BITS_0(0,b) = INBITS_0(0,b)$ for the first level of the first stage. As each full adder reduces three partial products to one of the same weight and one of the next higher weight, and each half adder converts two partial products to one of the same weight and one of the next higher, the resulting number of partial products in each following level can be written

$$\begin{aligned} BITS_j(k+1,b) = {} & BITS_j(k,b) + INBITS_j(k,b) \\ & - 2FA_j(k,b) - HA_j(k,b) + FA_j(k,b-1) + HA_j(k,b-1), \end{aligned} \tag{22}$$

for $k \in \{0, 1, 2, \ldots, K-1\}$, $b \in \{0, 1, 2, \ldots, w_{out}-1\}$, and with variables equal to zero for out-of-bounds arguments. The relations between the $BITS$ variables are depicted in Fig. 9 . Connections between the stages are defined by

Fig 9

$$BITS_{j+1}(0, b) = BITS_j(K, b). \tag{23}$$

Variables $REGS_j(b)$ denoting the number of pipeline registers for each stage are defined by

$$REGS_j(b) = BITS_j(K, b). \tag{24}$$

Thus, registers are added for all signals between the stages, as shown in Fig. 10 . The number of adders in each level is limited by the constraint

Fig 10

$$3FA_j(k, b) + 2HA_j(k, b) \leq BITS_j(k, b) + INBITS_j(k, b), \tag{25}$$

as the number of adder inputs can not exceed the number of partial products, for each level k and bit position b. Also, in order to utilize a VMA to sum the output, the number of output bits from the last stage is limited to 2 by the condition

$$BITS_{J-1}(K, b) + INBITS_{J-1}(K, b) \leq 2, \tag{26}$$

for $b \in \{0, 1, 2, \ldots, w_{out}-1\}$. Costs are defined for full adders, half adders and registers as C_{FA}, C_{HA}, and C_{REG}, respectively, and the filter structure is optimized by minimizing the sum

$$\begin{aligned} C = C_{\mathrm{FA}} \sum_{j=0}^{J-1} \sum_{k,b} FA_j(k, b) + C_{\mathrm{HA}} \sum_{j=0}^{J-1} \sum_{k,b} HA_j(k, b) + \\ + C_{\mathrm{REG}} \sum_{j=0}^{J-1} \sum_{b} REGS_j(b) \end{aligned} \tag{27}$$

The optimization problem as specified by (22)–(27) does not consider the length of the VMA. However, it may be possible to significantly reduce the length by introducing half adders to the least significant bits. The optimization problem can be modified to achieve a shorter VMA by adding a constraint to limit the number of output partial products to one for a number m of the least significant bits. This can be done by the constraint

$$BITS_{J-1}(K, b) + INBITS_{J-1}(K, b) \leq 1, \tag{28}$$

for $b \in \{0, 1, 2, \ldots, m-1\}$.

Whereas the problem as formulated is sufficient to find the optimal filter for a given architecture, the problem complexity can be significantly reduced by the addition of additional constraints. In particular, there will never be a reason, in terms of efficient reduction of the number of partial products, not to insert a full adder where at least three partial products are available. Hence, the number of full adders in a given position can be defined based on the number of partial products available as

$$FA_k(l, b) = \left\lfloor \frac{BITS_k(l, b) + INBITS_k(l, b)}{3} \right\rfloor, \tag{29}$$

which can be formulated as two linear constraints for each variable.

It should be noted that the optimization problem, as formulated, is independent of the coefficient representation, i.e., binary as well as any signed-digit representation may be used. However, if signed digits are used, either if the data is signed or if the coefficient contains negative digits, a constant term must be added to the sum, as discussed in Section II-A. As the placement of the term in the tree is arbitrary, the problem can be modified to insert the bits where they fit well. How to formulate the optimization problem to accomodate for the constant term is explained in IV-E. In IV-A to IV-D, the presented architectures are formulated as initial conditions for the optimization problem. In these formulations, the coefficient digits $h_{n,j}$ are defined as in (4) or (5), with $h_{n,j} \in \{0, 1\}$ for binary coefficients and $h_{n,j} \in \{-1, 0, 1\}$ for signed-digit coefficients.

A. DF1 architecture

For the DF1 architecture, all partial products are inserted in the first adder stage. The sum of the partial products is $\sum_{n=0}^{N} \sum_{j=0}^{w_d-1} \sum_{i=0}^{w_c-1} |h_{n,i}| \, 2^{i+j}$. Substituting $b = i + j$ and rearranging the sums allows the number of bitproducts of weight b to be written

$$INBITS_0(0, b) = \sum_{n=0}^{N} \sum_{j=0}^{w_d-1} |h_{n,b-j}| \,. \tag{30}$$

B. DF2 architecture

If the direct-form architecture is modified to utilize coefficient symmetry, the symmetry adders will add additional delay. Thus, the partial products involving bit 0 (the LSB) of the data are added in level 1, the partial products involving bit 1 of the data are added in level 2, and so on. Generally, the number of initial partial products in stage j and level k can be written

$$INBITS_j(k, b) = \sum_{n=0}^{(N+1)/2} |h_{n,b-Kj-k+1}| \tag{31}$$

for $1 \leq Kj + k \leq w_d - 1$, and

$$INBITS_j(k, b) = \sum_{n=0}^{(N+1)/2} (|h_{n,b-Kj-k+1}| + |h_{n,b-Kj-k}|) \tag{32}$$

for $Kj + k = w_d$.

C. DF3 architecture

For the partial symmetry case, the contributions of the different adders are separated. Assuming that a symmetry width of S adders is used, the partial products can be split into $\lceil w_d/S \rceil$ bins, where the first contains partial products from the S least significant data bits, the next bin contains partial products from the next S least significant data bits, and so on. Denoting the contribution from bin m by $B_j^m(k, b)$, the contributions for $m = 0, 1, 2, \ldots, \lceil w_d/S \rceil - 1$ can be written

$$B_j^m(k,b) = \sum_{n=0}^{(N+1)/2} |h_{n,b-Kj-k-mS+1}| \tag{33}$$

for $1 \leq Kj + k \leq w_d - 1$, and

$$B_j^m(k,b) = \sum_{n=0}^{(N+1)/2} (|h_{n,b-Kj-k-mS+1}| + |h_{n,b-Kj-k-mS}|) \tag{34}$$

for $Kj + k = w_d$. For $m = \lceil w_d/S \rceil$, the contribution can be written

$$B_j^m(k,b) = \sum_{n=0}^{(N+1)/2} |h_{n,b-Kj-k-mS+1}| \tag{35}$$

for $1 \leq Kj + k \leq (w_d \bmod S) - 1$, and

$$B_j^m(k,b) = \sum_{n=0}^{(N+1)/2} (|h_{n,b-Kj-k-mS+1}| + |h_{n,b-Kj-k-mS}|) \tag{36}$$

for $Kj + k = w_d \bmod S$. Finally, the combined contribution is the sum of the partial symmetry adder contributions

$$INBITS_j(k,b) = \sum_{m=0}^{\lceil w_d/S \rceil} B_j^m(k,b) \tag{37}$$

D. TF architecture

Denoting the polyphase factor by M, for the TF architecture the first M filter coefficient will be inserted in the last adder stage, the next M coefficients in the stage before, and so on. Thus, the number of initial partial products can be written

$$INBITS_{J-j-1}(0,b) = \sum_{n=Mj}^{M(j+1)-1} \sum_{t=0}^{w_d-1} |h_{n,b-t}| \,. \tag{38}$$

E. Constant term placement

If either the coefficient or the data contains digits with a negative sign, a constant compensation term must be added to the carry-save tree. However, these bits may be placed in an arbitrary stage, and in this section it is explained how the problem may be modified to place the bits optimally in terms of hardware resources. Define the constant, in two's complement representation, as

$$C = -c_{w_{out}-1} 2^{w_{out}-1} + \sum_{b=0}^{w_{out}-2} c_b 2^b, \tag{39}$$

where $c_b \in \{0,1\}$. Then define the ILP variables $CBITS_j(b) \in \{0,1\}$ for $j \in \{0,1,2,\ldots,J-1\}$, $b \in \{0,1,2,\ldots,w_{out}-1\}$, and add the constraint

$$\sum_{j=0}^{J-1} CBITS_j(b) = c_b \tag{40}$$

for $b \in \{0,1,2,\ldots,w_{out}-1\}$. Redefine (23) to

$$BITS_{j+1}(0,b) = BITS_j(K,b) + CBITS_{j+1}(b) \tag{41}$$

in order to add the constant bits to the carry-save tree.

V. Results

In this section, the different architectures are compared, and the choice of coefficient representation is investigated. For the energy and area estimations, a VHDL generator has been used to generate synthesizable VHDL code. The complete software package with ILP problem and VHDL code generator is available at `http://www.es.isy.liu.se/software/hsfir/`.

A. Architecture Comparison

Two filters have been used to evaluate the optimization algorithm, and the relative performance of the architectures. The filters are based on 4-tap and 16-tap moving average filters, $M = 4$ and $M = 16$, respectively. Both filters consist of three cascaded filters ($L = 3$). In all simulations, the numbers of full adders correspond to those given by (7) and (8), and the number of registers given by (16) and (18) were added to the optimized result for the DF1 and DF2 architectures, respectively. The filters were optimized using the ILP problem solver SCIP [2] with the costs $C_{\mathrm{FA}} = 3$, $C_{\mathrm{HA}} = 2$, and $C_{\mathrm{REG}} = 3$. Even though the area of a full adder is roughly twice that of a half adder, it was chosen to increase the half adder cost slightly as the routing associated to one full adder is likely less than that of two half adders. However, it should be noted that the optimization results seldom differ to those obtained using the more common costs of $C_{\mathrm{FA}} = 2$, $C_{\mathrm{HA}} = 1$, and $C_{\mathrm{REG}} = 2$.

The optimized filters have been compared with filters obtained using the Reduced Area [4] heuristic. The Reduced Area heuristic is claimed to minimize the number of registers when used in a pipelined multiplier reduction tree. However, it is interesting to note that this is in general not true for the bitproduct matrices resulting from filters implemented with carry-save adder trees. Especially for the TF architecture, the bit-level optimized adder trees may result in significantly reduced register usage, while also using fewer half adders.

Figure 11 shows the impact of the pipeline factor on the first filter with short wordlengths. For the 4-tap filter, $N_a = 1.8$, and according to (15) utilizing the coefficient symmetry does not lead to reduced arithmetic complexity for $w_d < 2.25$, and the DF2 architecture has thus not been included. Also, the half adder complexity was equal to 6 for all filters. It can be seen that the bit-level optimized filters use significantly less registers, especially for heavily pipelined implementations. It can also be seen that the TF architecture has a lower register complexity except for implementations with large stage height and one bit input. Fig 11

In Fig. 12 and Fig. 13 , respectively, the energy dissipation and active cell area (excluding routing) are shown. The area and energy measures are based on a 90 nm standard cell library using Synopsys Design Compiler. The energy estimations are obtained using a zero-delay model and assuming uncorrelated input data. In the considered application, using a zero-delay model is expected to yield relevant results as the amount of glitches is small due to the short critical paths. Also, as decimation filter for a sigma-delta ADC the assumption of uncorrelated input data is considered to be relevant. Fig 12 Fig 13

In Fig. 14 , the total cost of the optimized filters are shown. These include additional logic and arithmetic such as the algorithmic delays for the DF1 architecture and the adders used in the ripple-carry VMA, which Fig 14

are not considered in the optimization. By comparing Fig. 14 with Figs. 12 and 13, it can be concluded that the used cost function is a relevant measure for optimizing both energy dissipation and cell area. Whereas energy dissipation and cell area in general do not have a strong correlation, they can be expected to correlate well when the amount of glitches is small and uncorrelated input data is used. Thus, in the rest of this paper, only complexity results and energy dissipation results will be presented as cell area and cost are similar to energy dissipation.

In Fig. 15 , the implementation complexity of the 16-tap filter is shown, using one bit input data. It is apparent that the bit-level optimized filter for the TF architecture offers a lower register complexity, while also reducing the number of half adders. Fig 15

Figures 16 and 17 show the adder and register complexity, respectively, for increasing input wordlength w_d for the 4-tap filter. The stage height is $K = 2$. For the 4-tap filter, $N_a = 1.8$, and according to (15) the DF2 architecture has a lower arithmetic complexity for $w_d > 2$. However, even for $w_d = 6$, the reduction in arithmetic complexity is small compared to the increase in number of registers, as seen in Fig. 17. Energy dissipation estimations are shown in Fig. 18 , where also simulation results of the DF3 architecture have been included. Fig 16 Fig 17 Fig 18

B. Coefficient Representation

Different coefficient representations have been investigated for two filters shown in Fig. 19 . Using signed-digit coefficients yields a small decrease in energy dissipation. That the gain is not larger is because the additional constant vector is relatively large compared with the number of bit-products for such small filters. Also, the simulation has not taken into account the fact that adders with a constant bit input may be simplified. It can be expected that the difference between binary and MSD coefficients would increase as the data and/or coefficient wordlength increases. Fig 19

VI. Conclusion

In this paper, a method for bit-level optimization of pipelined carry-save reduction trees was proposed. The focus was on high-speed structures with a moderate number of partial products, such as those resulting from polyphase FIR decimation filters aimed at sigma-delta analog-to-digital converters. Typically in these applications, the wordlengths involved are small, and the filter coefficients are simple, limiting the number of partial products.

The algorithm was used to optimize filter realizations using both the direct-form and transposed direct-form architectures. For the direct-form architectures, utilizing the symmetry of the linear-phase coefficients was considered, and a form of partial symmetry limiting the number of pipeline registers for the symmetry adders was additionally considered. It was found that the transposed direct-form architecture provides the implementation with the lowest complexity in most cases, and that the register costs of the symmetry adders do not generally motivate the reduced arithmetic complexity achievable using the direct-form architecture.

The resulting costs of the optimized filters were compared with energy estimations, and it was concluded that the used costs are suitable for optimizing both energy dissipation and cell area. Also, it was shown how the optimization algorithm can be modified to allow for signed-digit coefficients, and simulations were provided showing the decrease in power dissipation for the transposed direct-form architecture.

ACKNOWLEDGEMENTS

The authors thank the anonymous reviewers for their valuable comments on the paper.

REFERENCES

[1] 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.

[2] T. Achterberg, "Constraint Integer Programming," Ph.D. dissertation, Technische Universität Berlin, 2007, http://opus.kobv.de/tuberlin/volltexte/2007/1611/.

[3] A. Berkeman, V. Öwall, and M. Torkelson, "A low logic depth complex multiplier using distributed arithmetic," *IEEE J. Solid-State Circuits*, vol. 35, no. 4, pp. 656–659, Apr. 2000.

[4] K. Bickerstaff, M. Schulte, and E.E., Swartzlander, Jr., "Reduced area multipliers," in *Proc. Int. Conf. Application-Specific Array Processors*, Nov. 1993, pp. 478–489.

[5] A. Blad and O. Gustafsson, "Bit-level optimized high-speed architectures for decimation filter applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2008.

[6] J. Candy, "Decimation for sigma delta modulation," *IEEE Trans. Commun.*, vol. 29, pp. 72–76, Jan. 1986.

[7] J. Candy and G. Temes, "Oversampling methods for A/D and D/A conversion," in *Oversampling Delta-Sigma Data Converters, pp. 1–25*. IEEE Press, 1992.

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

[9] Y. Dumonteix and H. Mehrez, "A family of redundant multipliers dedicated to fast computation for signal processing," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 5, May 2000, pp. 325–328.

[10] Y. Gao, L. Jia, J. Isoaho, and H. Tenhunen, "A comparison design of comb decimators for sigma-delta analog-to-digital converters," *Analog Integrated Circuits and Signal Processing*, vol. 22, no. 1, pp. 51–60, Jan. 2000.

[11] 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.

[12] E. B. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 29, pp. 155–162, Apr. 1981.

[13] K. Johansson, O. Gustafsson, and L. Wanhammar, "Implementation of elementary functions for logarithmic number systems," *Computers & Digital Techniques, IET*, vol. 2, no. 4, pp. 295–304, Jul. 2008.

[14] K.-Y. Khoo, Z. Yu, and Alan N. Willson, Jr., "Bit-level arithmetic optimization for carry-save additions," in *Proc. Computer-Aided Design, Digest of Technical Papers*, Nov. 1999, pp. 14–18.

[15] 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.

[16] U. Meyer-Baese, S. Rao, J. Ramírez, and A. García, "Cost-effective hogenauer cascaded integrator comb decimator filter design for custom ICs," *IEEE Electronics Letters*, vol. 41, no. 3, pp. 158–160, Feb. 2005.

[17] H. Ohlsson, B. Mesgarzadeh, K. Johansson, O. Gustafsson, P. Löwenborg, H. Johansson, and A. Alvandpour, "A 16 GSPS 0.18 μm CMOS decimator for single-bit $\Sigma\Delta$-modulation," in *Proc. IEEE Norchip*, November 2004, pp. 175–178.

[18] R. Schreier and G. C. Temes, *Understanding Delta-Sigma Data Converters*. John Wiley & Sons, NJ, 2005.

[19] J. E. Swartzlander, "Merged arithmetic," *IEEE Trans. Comput.*, vol. C-29, no. 10, pp. 946–950, Oct. 1980.

[20] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Eaglewood Cliffs, NJ: Prentice-Hall, 1993.

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

[22] L. Wanhammar and H. Johansson, *Digital Filters*. Linköping University, 2002.

Figure captions:

Fig. 1: Direct-form architecture (DF1).

Fig. 2: Direct-form architecture utilizing coefficient symmetry (DF2).

Fig. 3: Partial symmetry adder

Fig. 4: Transposed direct-form architecture (TF).

Fig. 5: Carry-save adder tree pipelined in J stages.

Fig. 6: One stage of a carry-save adder tree, with a maximum of K adders in the critical path.

Fig. 7: Resulting partial products when multiplying a three-bits signed data with 29 in (a) binary representation and (b) CSD representation. White dots corresponds to negated partial products.

Fig. 8: Pipelined ripple carry adder.

Fig. 9: Relations between the $BITS$ variables in the ILP problem.

Fig. 10: Relations between the stages in the ILP problem.

Fig. 11: Register complexity comparison of FIR CIC filters with $M = 4$ and $L = 3$.

Fig. 12: Energy dissipation estimations of FIR CIC filters with $M = 4$ and $L = 3$.

Fig. 13: Synthesized cell area of FIR CIC filters with $M = 4$ and $L = 3$.

Fig. 14: Cost of optimized FIR CIC filters with $M = 4$ and $L = 3$.

Fig. 15: Register/HA complexity of bit-level optimized FIR CIC filters with $M = 16$, $L = 3$, and $w_d = 1$. The number of half adders is 12 or 13 for all simulations.

Fig. 16: Adder complexity of FIR CIC filters with $M = 4$, $L = 3$, and $K = 2$.

Fig. 17: Register complexity of FIR CIC filters with $M = 4$, $L = 3$, and $K = 2$.

Fig. 18: Energy dissipation estimations of TF, DF1, DF2 and DF3 architectures with $M = 4$, $L = 3$, and $K = 2$.

Fig. 19: Binary and MSD coefficients for FIR CIC filters with $M = 4, L = 4$ and $M = 8, L = 3$.

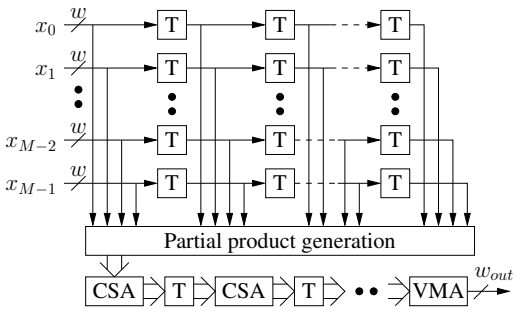

Fig. 1. Direct-form architecture (DF1).

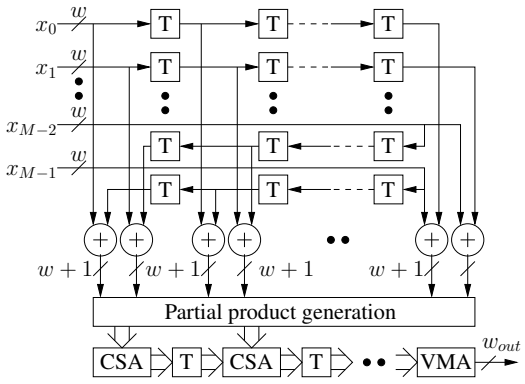

Fig. 2. Direct-form architecture utilizing coefficient symmetry (DF2).

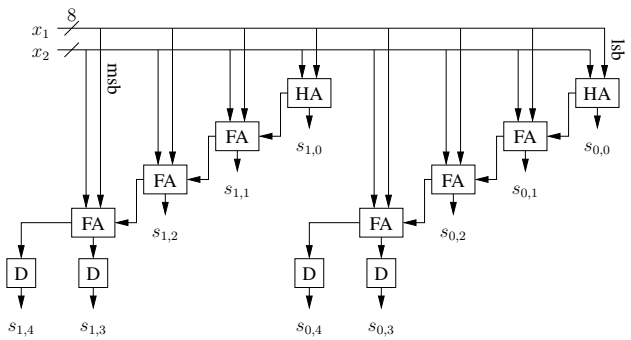

Fig. 3. Partial symmetry adder

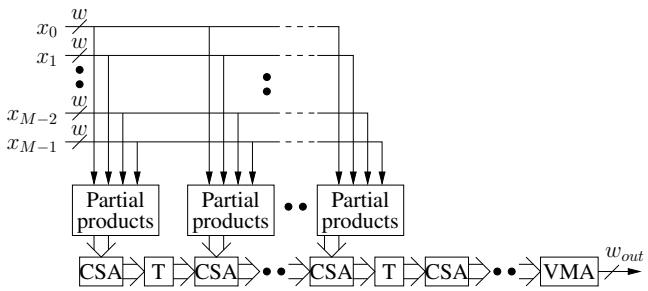

Fig. 4. Transposed direct-form architecture (TF).

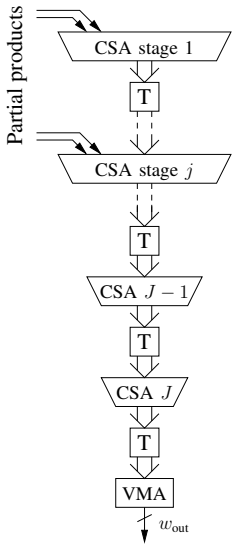

Fig. 5. Carry-save adder tree pipelined in J stages.

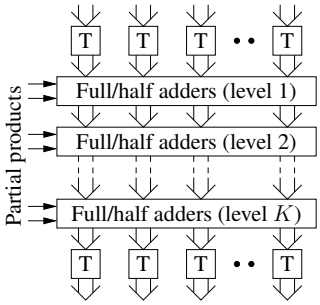

Fig. 6. One stage of a carry-save adder tree, with a maximum of K adders in the critical path.

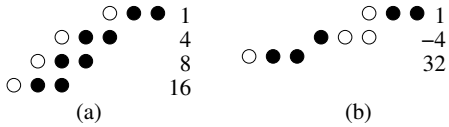

Fig. 7. Resulting partial products when multiplying a three-bits signed data with 29 in (a) binary representation and (b) CSD representation. White dots corresponds to negated partial products.

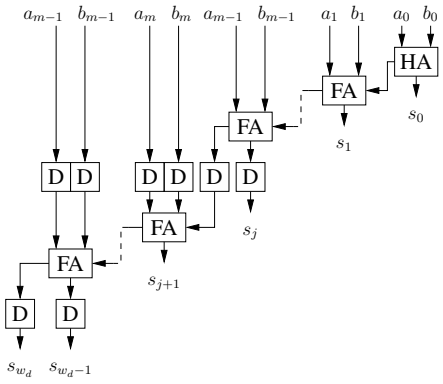

Fig. 8. Pipelined ripple carry adder.

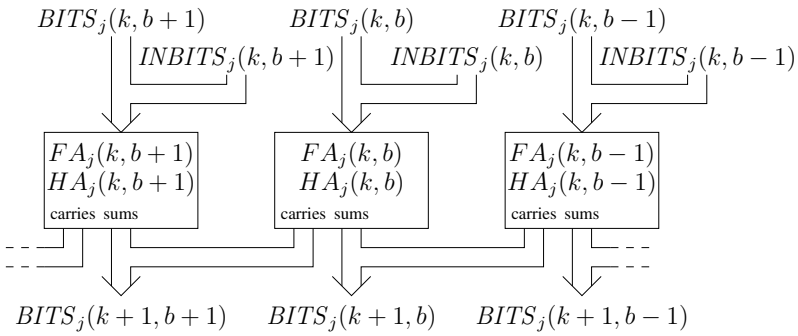

Fig. 9. Relations between the $BITS$ variables in the ILP problem.

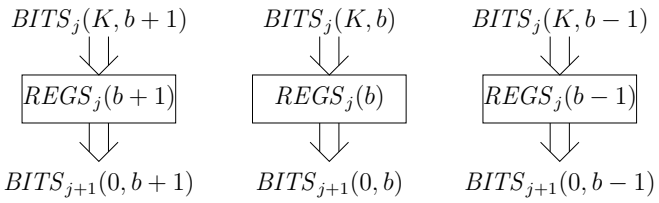

Fig. 10. Relations between the stages in the ILP problem.

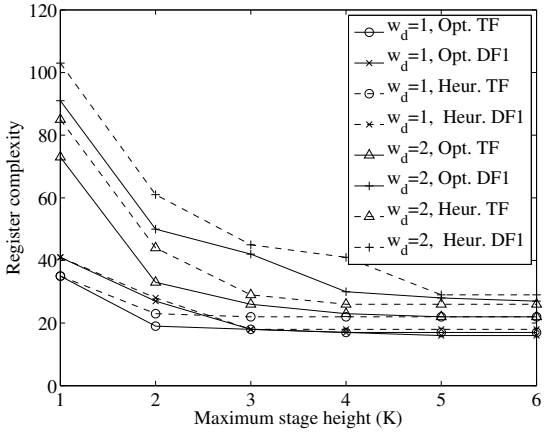

Fig. 11. Register complexity comparison of FIR CIC filters with $M = 4$ and $L = 3$.

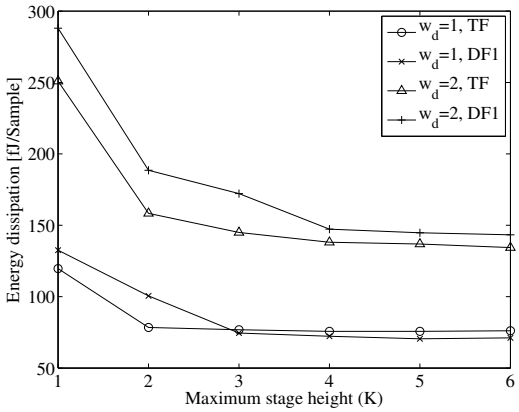

Fig. 12. Energy dissipation estimations of FIR CIC filters with $M = 4$ and $L = 3$.

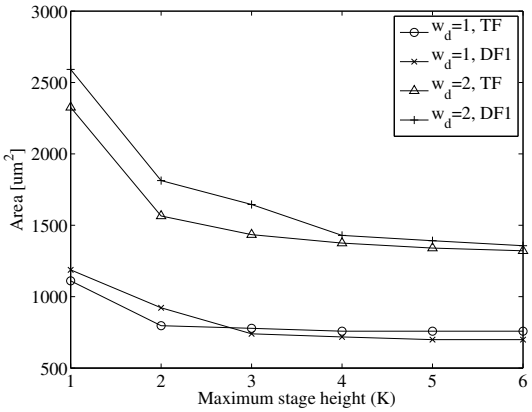

Fig. 13. Synthesized cell area of FIR CIC filters with $M = 4$ and $L = 3$.

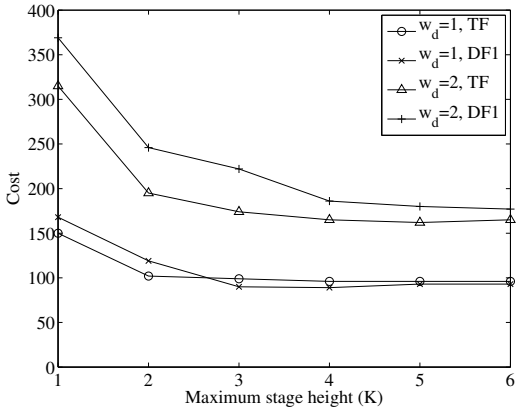

Fig. 14. Cost of optimized FIR CIC filters with $M = 4$ and $L = 3$.

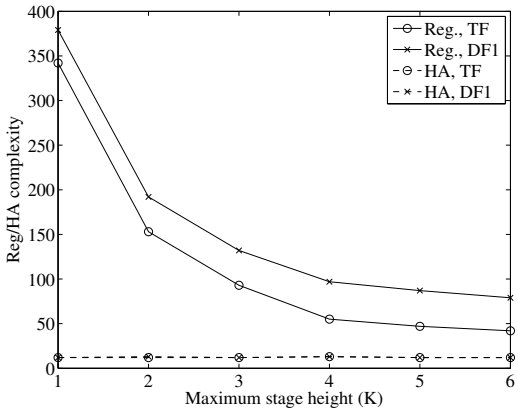

Fig. 15. Register/HA complexity of bit-level optimized FIR CIC filters with $M = 16$, $L = 3$, and $w_d = 1$. The number of half adders is 12 or 13 for all simulations.

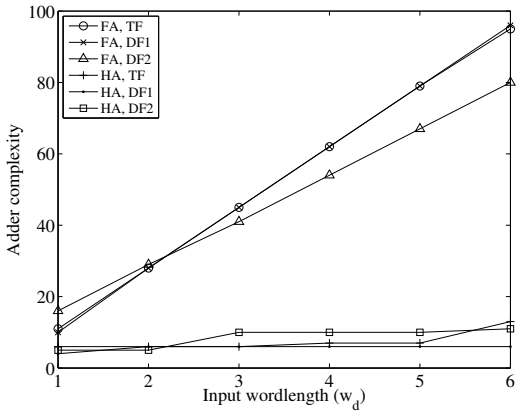

Fig. 16. Adder complexity of FIR CIC filters with $M = 4$, $L = 3$, and $K = 2$.

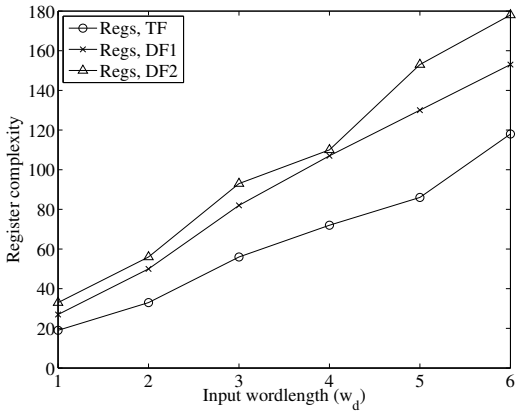

Fig. 17. Register complexity of FIR CIC filters with $M = 4$, $L = 3$, and $K = 2$.

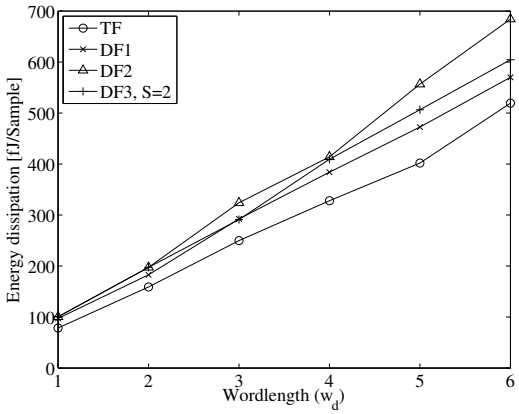

Fig. 18. Energy dissipation estimations of TF, DF1, DF2 and DF3 architectures with $M = 4$, $L = 3$, and $K = 2$.

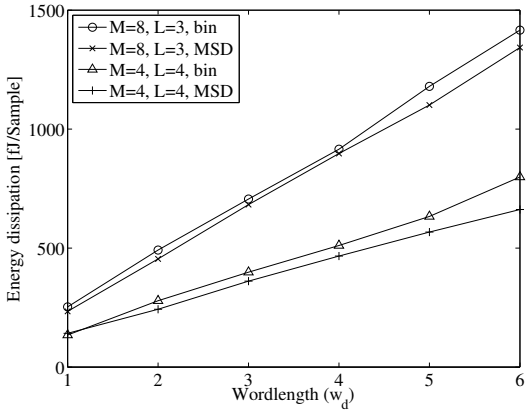

Fig. 19. Binary and MSD coefficients for FIR CIC filters with $M = 4, L = 4$ and $M = 8, L = 3$.