Optimization of Area in Digital FIR Filters using Gate-Level Metrics

Levent Aksoy
Istanbul Technical University
Department of Electronics Eng.
Maslak, Istanbul, Turkey
levent@ehb.itu.edu.tr

Eduardo Costa
Universidade Catolica de Pelotas
Rua Félix da Cunha, 412, Centro
Pelotas-RS, Brazil
ecosta@ucpel.tche.br

Paulo Flores Jose Monteiro
INESC-ID/IST
Rua Alves Redol, 9, 1000-029
Lisbon, Portugal
{pff, jcm}@inesc-id.pt

ABSTRACT

In the paper, we propose a new metric for the minimization of area in the generic problem of multiple constant multiplications, and demonstrate its effectiveness for digital FIR filters. Previous methods use the number of required additions or subtractions as a cost function. We make the observation that not all of these operations have the same design cost. In the proposed algorithm, a minimum area solution is obtained by considering area estimates for each operation. To this end, we introduce accurate hardware models for addition and subtraction operations in terms of gate-level metrics, under both signed and unsigned representations. Our algorithm not only computes the best design solution among those that have the same number of operations, but is also able to find better area solutions using a non-minimum number of operations. The results obtained by the proposed exact algorithm are compared with the results of the exact algorithm designed for the minimum number of operations on FIR filter instances and it is shown that the area of the design can be reduced by up to 18%.

Categories and Subject Descriptors

B.2.0 [**Arithmetic and Logic Structures**]: General.

General Terms

Algorithms, design.

Keywords

Multiple constant multiplication, FIR, area optimization.

1. INTRODUCTION

Finite impulse response (FIR) digital filters are widely used in digital signal processing by virtue of stability and easy implementation. The problem of designing FIR filters has received a significant amount of attention during the last decade, as the filters require a large number of multiplications, leading to excessive area, delay, and power consumption even if implemented in a full custom integrated circuit. Previous works have focused on the design of filters with minimum area by replacing the multiplication operations with constant coefficients by addition, subtraction, and shifting operations. Since shifts are free in terms of hardware, the design problem can be defined as the minimization of the number of addition/subtraction operations to implement the coefficient multiplications. This is in fact applicable to several other problems in digital signal processing and is generically known as the multiple constants multiplication (MCM) problem.

There have been a number of algorithms proposed for the optimization of the number of operations. These methods range from the graph based coefficient synthesis techniques [1, 2] and exhaustive enumeration of all possible digit patterns [7] to the sharing of common digits [4]. To further reduce the complexity of the design, the coefficients can be expressed in canonical sign digit (CSD) or represented in minimal sign digit (MSD). Both representations use the signed digit system with the digit set $\{\overline{1},0,1\}$, where $\overline{1}$ denotes -1, and have a common property that the number of non-zero digits is minimal. The CSD representation provides a unique representation for every constant, since two non-zero digits are not adjacent in CSD. A constant can have several MSD representations, because non-zero digits can be consecutive in MSD. In [6], it is shown that using MSD representation yields better solutions than CSD in the optimization of the number of operations, since it has the same number of non-zero digits as CSD, but provides multiple alternative representations for a constant. Recently, we have proposed an exact algorithm that maximizes the sharing of partial terms using 0-1 Integer Linear Programming (ILP) [3].

Although these heuristic and exact algorithms find best solutions for the optimization of the number of operations, these solutions may prove sub-optimal when implemented at the gate level. In this work, we consider gate-level metrics in the selection of an operation to be implemented. We introduce architectures based on half adders (HAs) and full adders (FAs) for addition and subtraction operations under unsigned and signed input models. We implemented an exact algorithm that is based on the exact algorithm of [3], but it is modified in order to take into account the gate-level metrics of the operations. In the proposed algorithm, the objective function is defined as a linear combination of optimization variables representing operations with their cost values used in the synthesis environment. Note that the area of an operation implemented at the gate-level depends on:

- the type of the operation (addition or subtraction),
- the shifted input (minuend or subtrahend) in a subtraction,
- the number of shifts at the inputs,
- the position of the operation in the architecture (that influences the number of bits),
- the range and type of numbers considered (unsigned or signed).

The rest of the paper is organized as follows. In Section 2, the exact algorithm is described. The cost of addition and subtraction operations, signed and unsigned, considering gate-level metrics is derived in Section 3. Experimental results are given in Section 4 and the paper concludes with Section 5.

2. THE EXACT ALGORITHM

In this section, we present the implementation of the exact algorithm designed for the minimization of area and describe the Booelan network constructed by the algorithm that represents the optimization problem. The implemented algorithm can be used for any type of coefficient representation: binary, CSD, or MSD. Initially, we describe the MSD implementation of the algorithm and then, we summarize the changes for binary and CSD representations.

In the preprocessing phase of the algorithm, we convert all filter coefficients to positive and then make odd by successive divisions by 2. Each new resulting coefficient is added to a set called $Iset$ that includes the minimum number of coefficients necessary to be synthesized. For each element i in $Iset$, all MSD representations are determined using $\lceil \log_2(i) \rceil + 1$ bits and inserted in $Cset$. Although $Cset$ begins with all the MSD representations of the coefficients, $Cset$ will be augmented with MSD representations of partial terms during the execution of our algorithm. In the main algorithm loop, an element c, representing a number i, is removed from $Cset$ and is processed to determine its covers in the following way: 1) compute all partial term pairs that covers the element c; 2) convert each element of the cover pair to positive and make it odd; 3) add each cover pair to the corresponding set of covers of the element being processed, $Aset_i$; 4) add the MSD representations of each cover to $Cset$, if the representation has not been processed yet and it is not already in $Cset$. Covers with only one non-zero digit are skipped. This loop is repeated until there are no more elements in $Cset$. The pair of elements in each $Aset_i$ represents all possible implementations of a value i based on its MSD representations.

The final 0-1 ILP optimization model is generated in three steps: 1) for each pair element in $Aset_i$, generate the corresponding AND gate, with an additional input that represents an optimization variable for the AND gate. Generate an OR gate for the value i with the outputs of all the ANDs resulting from $Aset_i$; 2) identify all the OR gate outputs that represent a coefficient (values belonging to $Iset$) and set their outputs to value 1; 3) generate the objective function to be minimized. This function is a linear combination of the optimization variables at inputs of the AND gates.

As described in the first step, the network constructed by the algorithm only includes AND and OR gates. In this network, an AND gate represents an addition/subtraction operation and an OR gate combines the possible ways of implementation of a filter coefficient or a partial term. The primary inputs of the network represent the filter input or its shifted versions. The primary outputs of the network are the OR gate outputs that generate the filter coefficients. The number of inputs for each AND gate is three: two are either primary inputs or OR gate outputs and the third is an optimization variable. The inputs of an OR gate are the outputs of AND gates associated with the partial term. As an example, suppose that the value 51 ($10\bar{1}010\bar{1}$ in CSD) is a coefficient of the filter to be synthesized. The Boolean network generated by the exact algorithm is given in Figure 1 with the elimination of 1-input OR gates for 3, 17, and 63 partial terms. The type of each operation (adder or subtracter) is given inside of each AND gate. Observe that the coefficient 51 can be implemented in 7 different ways including additions and subtractions.

Note that the algorithm can be easily adapted to obtain the 0-1 ILP optimization model with different coefficient representations. In this case, the $Cset$ starts with the binary or CSD representations of the filter coefficients.

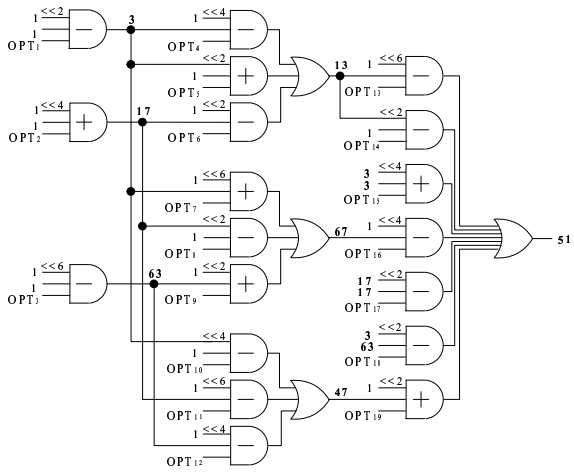

Figure 1: The network for the coefficient 51 in CSD.

In [3], since the maximization of the partial term sharing is realized, each optimization variable represents a partial term and its cost value is assigned to 1. In our algorithm, since we focus on the minimization of the area in a digital filter, an optimization variable is assigned to each operation and the cost value of each optimization variable in the objective function is determined using gate-level metrics as described in Section 3. Note that an exact algorithm for the minimization of the number of operations can be designed, when the cost value of each optimization variable, representing an operation, is assigned to 1. To obtain an exact solution, we use an efficient SAT-based 0-1 ILP solver [5] that incorporates several advanced optimization techniques and has been applied to several classes of problems.

3. THE PROPOSED MODELS

In this section, we describe the implementation of addition and subtraction operations in the design and determine the cost of each operation in terms of HAs, FAs and logic gates in a given technology library. Since the shifts are free in terms of hardware, the filter coefficients and partial terms are considered as odd numbers. There are three different types of operations that we can consider:

$A_{\ll S_A} + B_{\ll S_B}$ (an adder where $S_A = 0$, $S_B = S$)
$A_{\ll S_A} - B_{\ll S_B}$ (a subtracter where $S_A = S$, $S_B = 0$)
$A_{\ll S_A} - B_{\ll S_B}$ (a subtracter where $S_A = 0$, $S_B = S$)

In given operations, A and B represent the numbers at the inputs of the operation, S_A and S_B denote the number of shifts on the numbers A and B respectively. The following parameters are considered during the computation of each operation cost.
S: the number of shifts
n_A: the number of bits of A in the input
n_B: the number of bits of B in the input
n_m: minimum number of bits: $min(n_A + S_A, n_B + S_B)$
n_M: maximum number of bits: $max(n_A + S_A, n_B + S_B)$

We note that the number of the bits at the inputs of an operation depends on the bit width of the filter input that is

Table 1: Implementation cost of addition and subtraction operations (left) and experimental data (right).

Operation	$A+B_{\ll S}$		$A_{\ll S}-B$		$A-B_{\ll S}$	
Parameter	Unsigned	Signed	Unsigned	Signed	Unsigned	Signed
#FA	n_m-S-1	n_M-S-1	n_B-S	n_A	n_B-1	n_M-S-1
#HA	n_M-n_m+1	1	$S-1$	$S-1$	0	0
#HA′	0	0	n_A+S-n_B	0	n_A-n_B-S+1	1
#inv	0	0	n_B	n_B	n_B	n_B

Area (μm^2)	Max Delay (ns)	Delay Carry (ns)
58	0.261	0.194
32	0.185	0.137
35	0.185	0.085
6	0.06	–

denoted by N and the inputs that are represented by the filter coefficients or partial terms. The cost of each operation is determined considering unsigned and signed numbers, since these lead to different implementations due to the sign extension and is formulated by considering the overlap between inputs and taking into account specific cases.

3.1 Addition operation $A+B_{\ll S}$

The implementation cost of an addition operation in terms of HAs and FAs are given in Table 1. In Figure 2, examples on unsigned input models are given. Observe that larger number of shifts at the input achieves smaller area, since shifts are implemented with only wires in the design. Note that when the number of FAs is negative in the unsigned input case, no hardware is needed and the operation can be implemented with only wires as illustrated in the second example. This situation occurs, when the number of shifts of the operand B is equal to or higher than the number of bits of the operand A. In the signed input case, this situation never occurs, due to the sign extension of the operand A.

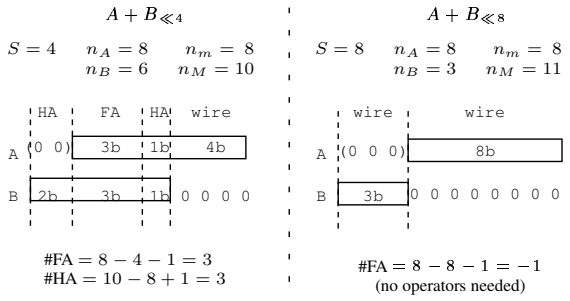

Figure 2: Examples on the computation of the cost of an $A+B_{\ll S}$ operation under unsigned input.

3.2 Subtraction operations $A_{\ll S}-B$ and $A-B_{\ll S}$

A subtraction operation is implemented using 2's complement, i.e., $A+\overline{B}+1$. So, the inverter (inv) is included in the cost of subtraction operations. Additionally, a different type of HA block denoted by HA′ is introduced. HA′ block is the special implementation of FA block when one of the inputs is 1 (as opposed to HA, another special implementation of FA when one of the inputs is 0). Suppose the input B_i is 1, the addition (sum) and carry output ($Cout$) are the functions of the input A_i and the carry input (Cin) given as $sum=\overline{Cin\oplus A_i}$ and $Cout=Cin+A_i$. The implementation cost of these subtraction operations are given in Table 1. In Figure 3, examples on unsigned input models are given for both subtraction operations. For the $A_{\ll S}-B$ operation, observe that the inverter accounted for the first bit is required not to generate the first bit of the result (since this is always equal to the first bit of the operand B), but to generate the carry taken to the second bit. For the $A-B_{\ll S}$ operation, we note that the shifts can be fully utilized by starting addition with the first digit of the inverted operand B resulting in a smaller area. Observe that the cost of the operation is computed without HA blocks as opposed to the $A_{\ll S}-B$ subtraction operation.

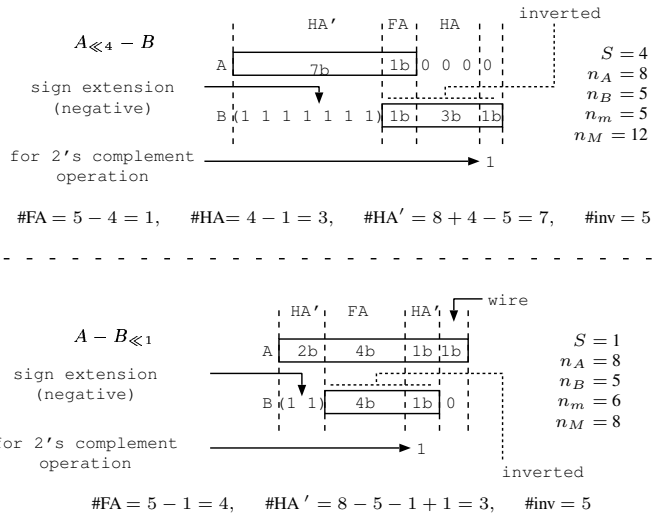

Figure 3: Examples on the computation of the cost of subtraction operations under unsigned input.

4. EXPERIMENTAL RESULTS

In this section, we present the results that are obtained with the minimum number of operations and the minimum area objectives. The data associated with the HA, HA′, FA blocks and an inverter were taken from UMC Logic 0.18μm Generic II library and are given in Table 1, on the right. We note that while *inv*, *HA*, and *FA* are primitive gates of the library, *HA′* was implemented using the gates in the library as defined in Section 3.2.

As an experiment set, we used filter instances where the coefficients were computed with MATLAB using the Remez algorithm. The specifications of filters are given in Table 2 where *pass* and *stop* are the normalized passband and stopband frequencies respectively; *#tap* is the number of coefficients; and *width* is the bit-width of the coefficients. The filter coefficients are defined in MSD representation, since MSD gives more promising solutions than binary and CSD.

The exact solutions with minimum number of operations and minimum area objectives are obtained on filter instances

Table 2: Filter specifications.

Filter	*pass*	*stop*	*#tap*	*width*
1	0.20	0.25	120	8
2	0.10	0.25	100	10
3	0.15	0.25	40	12
4	0.20	0.25	80	12
5	0.24	0.25	120	12
6	0.15	0.25	60	14
7	0.15	0.20	60	14
8	0.15	0.20	100	16
9	0.10	0.15	60	14
10	0.10	0.15	100	16

Table 3: Experimental results on unsigned (top) and signed (bottom) models.

N	8 bits						16 bits						24 bits					
Objective	# Operations			Area			# Operations			Area			# Operations			Area		
Filter	oper	area	delay	oper	area	delay	oper	area	delay	oper	area	delay	oper	area	delay	oper	area	delay
1	10	5114	4.2	10	4477	4.9	10	9994	7.3	10	9213	9.6	10	14874	10.4	10	13949	14.3
2	18	9465	7.1	18	7898	5.7	18	18249	11.7	18	16394	10.3	18	27033	16.4	18	24890	15.0
3	16	8847	6.7	17	8407	5.8	16	16751	11.4	16	16218	10.4	16	24655	16.0	16	23978	15.1
4	29	16462	6.9	29	13537	6.0	29	30638	11.2	29	27410	12.2	29	44814	16.2	29	41202	18.4
5	34	17645	7.1	34	15310	6.6	34	33949	11.7	34	31326	11.3	34	50253	16.4	34	47342	15.9
6	22	12673	8.4	23	12008	7.3	22	23457	14.6	22	22816	12.4	22	34241	20.8	22	33456	17.1
7	34	20310	7.5	35	17518	7.0	34	36854	12.2	34	33871	12.2	34	53398	16.8	34	50162	16.4
8	47	27798	7.8	51	24092	7.3	47	51003	14.1	47	47901	12.8	47	73819	20.3	47	70311	19.0
9	33	20787	8.2	35	16998	6.2	33	37140	14.4	35	34134	10.8	33	53316	20.6	34	50769	16.2
10	49	29474	8.6	54	25497	7.6	49	53714	15.6	49	50145	15.5	49	77458	21.8	49	73649	21.7
Avg.(%)	100	100	100	104.8	86.5	88.9	100	100	100	100.7	92.8	94.7	100	100	100	100.3	94.7	96.2
Min.(%)	–	–	–	100.0	81.8	75.5	–	–	–	100.0	89.5	75.1	–	–	–	100.0	91.9	78.7
Max.(%)	–	–	–	110.2	95.0	117.3	–	–	–	106.1	97.3	131.7	–	–	–	103.0	97.7	136.9
1	10	5924	4.6	10	5152	4.1	10	10804	7.7	10	9888	7.2	10	15684	10.8	10	14624	10.3
2	18	11098	7.8	18	9050	5.7	18	19882	12.5	18	17546	10.4	18	28666	17.1	18	26042	15.1
3	16	10552	7.3	16	10054	6.1	16	18456	12.0	16	17814	10.8	16	26360	16.6	16	25574	15.4
4	29	19333	7.5	29	15944	6.2	29	33509	12.1	29	29736	12.5	29	47685	16.8	29	43528	18.7
5	34	20682	7.9	34	17884	6.7	34	36986	12.5	34	33900	11.3	34	53290	17.2	34	49916	16.0
6	22	15396	9.4	23	14721	8.0	22	26180	15.6	22	25612	13.3	22	36964	21.9	22	36252	18.0
7	34	24625	8.5	35	21248	7.3	34	41169	13.2	34	37762	13.2	34	57713	17.8	34	54066	17.8
8	47	34149	9.2	50	29950	8.5	47	56965	15.4	47	52609	13.4	47	79781	21.6	47	74993	19.6
9	33	24375	9.2	36	21216	6.7	33	40551	15.4	34	37822	12.1	33	56727	21.6	34	54078	16.8
10	49	36280	10.5	52	31837	8.5	49	60024	16.7	49	55719	16.2	49	83768	22.9	49	79223	22.4
Avg.(%)	100	100	100	103.8	87.5	82.9	100	100	100	100.3	92.4	90.5	100	100	100	100.3	94.2	92.3
Min.(%)	–	–	–	100.0	81.5	72.8	–	–	–	100.0	88.3	78.9	–	–	–	100.0	90.8	77.8
Max.(%)	–	–	–	109.1	95.6	93.3	–	–	–	103.0	96.5	102.6	–	–	–	103.0	98.1	111.1

Table 4: Effect of the bit widths of filter input over area on unsigned input model.

Filter	D_8^8	D_{16}^8	D_{24}^8	D_8^{16}	D_{16}^{16}	D_{24}^{16}	D_8^{24}	D_{16}^{24}	D_{24}^{24}
Avg. (%)	100	102.1	102.7	101.7	100	100.2	102.3	100.02	100
Min. (%)	–	100.0	100.0	100.0	–	100.0	100.0	100.0	–
Max. (%)	–	105.7	106.0	103.9	–	101.1	105.1	100.1	–

for unsigned and signed input models, when the bit widths of the filter inputs, i.e., N, are 8, 16, and 24. The results on unsigned and signed models are given in Table 3, top half and bottom half respectively. In this table, *oper* denotes the number of operations, and *area* and *delay* denote the total area and the delay of the multiplier block of the filter, respectively.

In this experiment, we observe that a minimum solution in terms of the number of operations may not give the minimum area design (e.g., Filter 10 with 8 bits), and the area of the design can be improved using the proposed approach. As can be seen from experimental results, the reduction in area can be 13.5% and 12.5% on average for unsigned and signed models respectively. Since the optimization of area yields the optimization of the number of half and full adders, the delay of the design is decreased on most of the filter instances.

As discussed in the previous section, the cost of each operation is a function of the input bit-width, N. In order to analyse the impact of this parameter, we first computed the set of operations that lead to the best area solutions using a given value for N (represented as D_N^N, and this is optimized bit-width, *opt*). We then used this set of operations in the implementation of a filter with a different input bit-width (value *imp*) and obtained its area, the value D_{opt}^{imp}. Table 4 presents the maximum, minimum and average difference between the area D_{opt}^{imp} and D_{opt}^{opt} $(opt, imp \in \{8, 16, 24\})$, giving us a measure how much we loose by not using the correct value of N. In this experiment, we observed that parameter N affects the minimum area solution only slightly, with an average penalty between 2% and 3% and a maximum penalty of 6%, over our set of instances.

5. CONCLUSIONS

In this work, an exact algorithm that minimizes the area of the multiplier block of a digital filter is proposed. The area estimate of each operation is defined in terms of gate-level metrics and is used in the objective function to weight the cost of the operation. We present results indicating that if these issues are ignored, i.e., if the objective function is limited to the number of operations, the actual hardware implementation can be far from optimum. Although these results focus on FIR filters, our method can be directly applied to any system that uses multiple constant multiplications.

6. REFERENCES

[1] D. R. Bull and D. H. Horrocks. Primitive Operator Digital Filters. *IEE Proceedings G*, 138(3):401–412, 1991.

[2] A. Dempster and M. Macleod. Use of minimum-adder multiplier blocks in FIR digital filters. *IEEE TCAS II*, 42(8):569–577, 1995.

[3] P. Flores, J. Monteiro, and E. Costa. An Exact Algorithm for the Maximal Sharing of Partial Terms in Multiple Constant Multiplications. In *Proc. of ICCAD*, pages 13–16, 2005.

[4] R. Hartley. Subexpression Sharing in Filters using Canonic Signed Digit Multipliers. *IEEE TCAS II*, 43(10):677–688, 1996.

[5] V. Manquinho and J. Marques-Silva. Effective Lower Bounding Techniques for Pseudo-Boolean Optimization. In *Proc. of DATE*, pages 660–665, 2005.

[6] I.-C. Park and H.-J. Kang. Digital Filter Synthesis Based on Minimal Signed Digit Representation. In *Proc. of DAC*, pages 468–473, 2001.

[7] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova. A New Algorithm for Elimination of Common Subexpressions. *IEEE TCAD*, 18(1):58–68, 1999.