Optimization Method for Broadband Modem FIR Filter Design using Common Subexpression Elimination

Robert Pasko [*], Patrick Schaumont [**], Veerle Derudder [**], Daniela Durackova [*]
[*] Faculty of Electrical Engineering & Information Technology
Dept. of Microelectronics
Bratislava, Slovakia
[**] IMEC Leuven
Belgium

Abstract - An approach for broadband modem FIR filter design optimization is presented. It addresses the minimization of the number of adder-subtractors used in the hardware implementation of a FIR filter (Multiple Constant Multiplication problem). The method is based on identification and elimination of n-bit pattern common subexpressions in a set of filter coefficients by means of an exhaustive search. We give an algorithm description of our solution and demonstrate the performance on selected examples. A comparison of the results obtained by other authors is made and finally optimization and synthesis results on a realistic example, a 64-tap root-raised-cosine filter with 10 bit CSD coefficients is given.

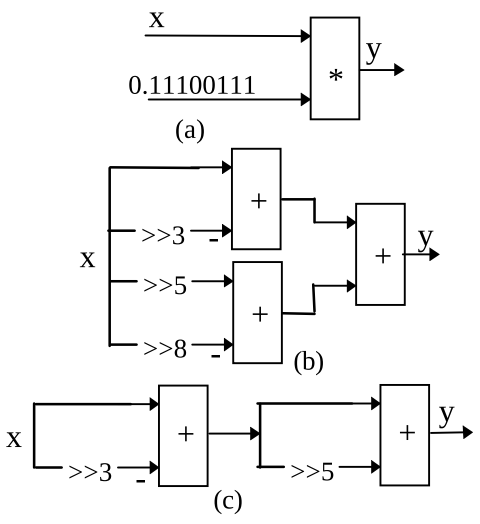
I. Introduction

The advent of broadband modems in consumer applications such as access network communications has attracted a lot of research towards the design of high speed FIR filters. The design requirements for these filters are such that they must process wideband signals with considerable accuracy. Because of the speed requirements, programmable solutions such as DSP cores are of no use in dealing with these problems. Rather, an application specific approach in hardware is needed. Since the existing standardization efforts [1,2] specify only the shape of the filter responses, and make no statement regarding their implementation, efficient VLSI synthesis methods are needed.

In this paper, an algorithm for optimization of a transpose form FIR filter with constant coefficients by an exhaustive search is presented. Today, exhaustive search (or brute force) algorithms are often combined, or even completely replaced, by sophisticated heuristic methods to gain the performance required in complex tasks, e.g. chess game algorithms. On the other hand, assumed appropriate algorithms are used, exhaustive search can be used to solve some less complex problems with very satisfactory results. We have used an exhaustive search method to find multiple common bit patterns in a FIR filter coefficient array. By removing multiple occurrences of the same pattern, and calculating these only once, hardware is saved. For larger filters, this leads to important area reductions, as will be demonstrated. During the optimization process, several techniques are used to obtain a cheaper implementation. First, all multiplications in a FIR filter are splitted into sequences of add-shift operations using a radix-2 signed digit representation of the coefficients. A Canonic Signed Digit (CSD) representation [4] is used. It is a representation with a minimum number of non-zero bits, and therefore a minimimum number of adder-subtractors are nexessary for the expansion.

In the next step, the set of coefficients to implement is subject to Common Subexpression Elimination (CSE). This involves finding multiple bit-patterns in the set of coefficients and coding these repetitions as common subexpressions into the filter structure. Hardware then is saved by the reduction of the number of adders necessary. An example of CSE optimization is demonstrated in Figure 1.

.



(a) multiplication with coeff. 0.11100111

(b) CSD add-shift expansion (1.00$\underline{1}$0100$\underline{1}$)

(c) add-shift after 100$\underline{1}$ pattern elimination

Figure 1. CSE optimization example

Common subexpression elimination has been already examined in various papers [5-8]. We briefly indicate the approaches that are followed. In [5], iterative elimination of only 2-bit common subexpressions is proposed, in [6] an iterative matching algorithm is used, in [7] a matrix-based approach for subexpression sharing is presented and in [8] a reduced adder graph algorithm is described. Some of these algorithms use heuristics to derive a cheaper implementation. In our method, we use an exhaustive search to find the common subexpressions in filter coefficients. Thanks to an efficient coding of the search algorithm, a better optimization is obtained in combination with efficient runtimes.

II. Method Overview

The basic steps of our method are shown in Figure 2. As input, a set of filter coefficients in floating point representation is used. During the first step, each of these coefficients is converted into the closest CSD representation according to user-selected parameters. This step is examined in section III.
Next, the CSD coefficients set is sent to CSE. During this step, multiple bit-patterns are eliminated from the set. Based on the reduced specification, VHDL behavioral code of the FIR filter is generated. CSE is discussed in detail in section IV. The performance of our algorithm is demonstrated with two examples in section V.1. In the first one, different coefficient sets are optimized and ratios between the number of adders-subtractors in initial and optimized sets are calculated. In the second example, a Root-Raised-Cosine 64-tap FIR filter with 10bit CSD coefficients is synthesized. Section V.2 then compares the obtained results to that of other authors. Lastly, conclusions are drawn in section VI.

III. Quantization and CSD Coding

The initial specification for the transposed-form FIR filter is given as a set of floating point coefficients, each representing one tap coefficient. The first task towards implementation is to quantize these into finite wordlength CSD representation. The number of bits required depends on the application at hand. For the case of modem design, some design rules can be found in [4]. A canonic signed digit representation is defined as a representation with a minimal number of non-zero digits for which no two non-zero digits are adjacent. The efficiency of CSD representation for constant multiplication is demonstrated in Figure 1(b)., where only 3 adders instead of 5 are necessary. A Floating point coefficient is converted during quantization to its closest CSD representation according to three criterions : the coefficient wordlength, the maximum number of non-zero bits, and the allowed quantization error. No other optimization is performed during this step, though several optimization methods are discussed in [4].

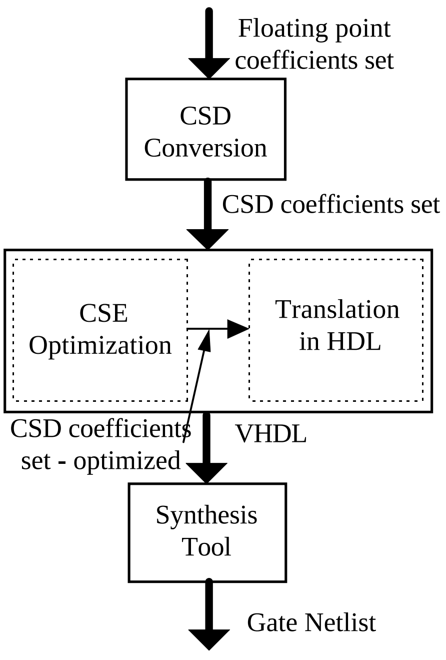


Figure 2. Method flow-graph

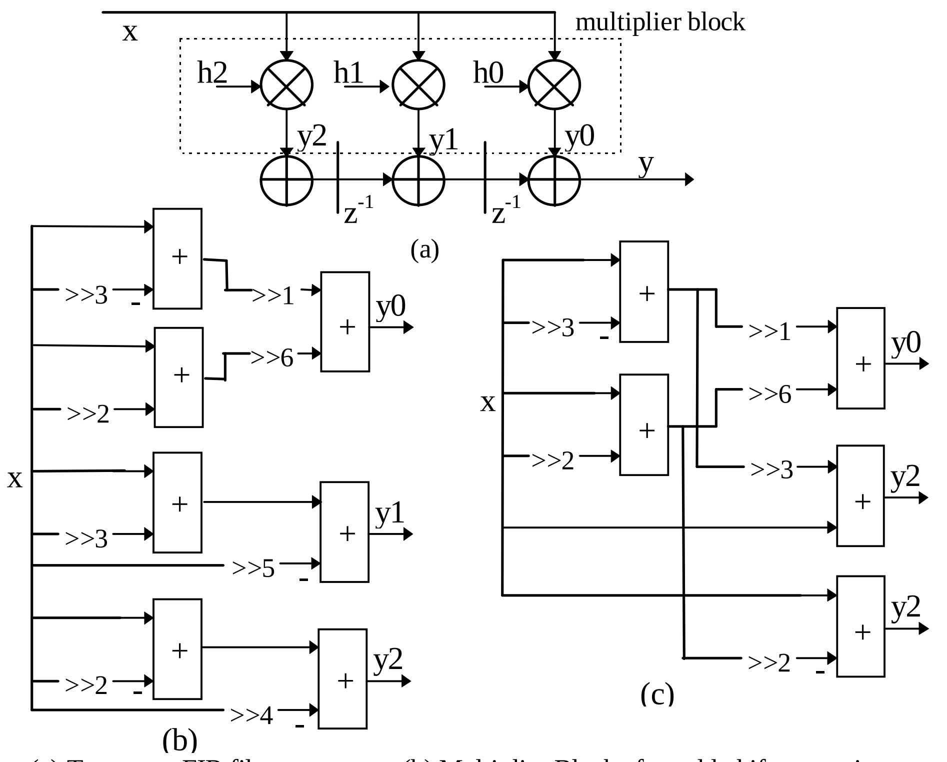
IV. Common Subexpression Elimination in FIR Filters

During CSE optimization, redundant hardware is minimized by finding repeating structures and implementing them only once. A demonstration example of CSE is shown in Figure 3. A 3-tap transpose-form FIR filter is depicted in Figure 3.a, with coefficients in Table 1.

Figure 3(b) shows the implementation after CSD add shift expansion. In this implementation, the patterns $100\underline{1}$ and 101 both occur twice, and this is detected by our algorithm. The optimized structure is shown in Figure 3(c). We indicate that the relative position of a bit pattern is not of importance since all shifts are considered hardwired.

h_0	$0.100\underline{1}0101_B$
h_1	$1.00100\underline{1}00_B$
h_2	$1.0\underline{1}0\underline{1}0000_B$

Table 1. Filter Coefficients in filter 1.



(a) Transpose FIR filter structure (b) Multiplier Block after add-shift expansion
(c) Hardware reduction after CSE optimization

Figure 3. CSE example

During CSE, an exhaustive search for common n-bit patterns is performed. We therefore need to examine all possible common subexpressions in the pattern set.
To do this, the frequency of each subexpression pattern is determined, and next the most common pattern is chosen and eliminated. This process is repeated until there does exist no longer a coefficient with frequency higher than 1. The complete block-diagram of our algorithm is presented in Figure 4. An parameter n is required, which must be in the range <2, (N+1)/2>. The upper bound is equal the to maximum number of non-zero bits in a N-bit CSD number.
During statistics creation, all possible n-bit patterns of each coefficient are identified and included into the frequency statistics. Next, the set of frequency statistics is searched for the most common pattern and, assumed a frequency is higher than one, all occurrences of the selected pattern are eliminated. Afterwards, the selected pattern is added as a new coefficient at the end of the coefficient set. Also, because elimination of one pattern must influence the frequency statistics of the others, the statistics must be re-evaluated. This procedure is repeated for *n*, *n-1*, *n-2 ... 2* bit patterns.
We will now discuss the important steps of the optimization procedure in more detail. These include :
1. Pattern identification
2. Pattern selection
3. Searching algorithm

IV.1 Pattern Identification

In an exhaustive search, all possible combinations of n-non-zero bit patterns in a coefficient must be examined. Since a pattern can only be eliminated once, we must also detect the occurrence of the same patterns within each other. For example, the valid 2-non-zero bit patterns of coefficient 010$\underline{1}$0$\underline{1}$0$\underline{1}$ are in the Table 2. The frequency of coefficient $\underline{1}$0$\underline{1}$ is one, because the second instance contains a second bit from the first one. Assuming that, for example pattern $\underline{1}$000$\underline{1}$ will be identified as the most frequent for example, then the original coefficient will be replaced with 01000$\underline{1}$00 and pattern $\underline{1}$000$\underline{1}$ will be added to the coefficient set as a new coefficient.

Bit pattern	**Frequency**
10$\underline{1}$	1
1000$\underline{1}$	1
100000$\underline{1}$	1
$\underline{1}$0$\underline{1}$	1
$\underline{1}$000$\underline{1}$	1

Table 2. 2 non-zero-bit patterns of coefficient 010$\underline{1}$0$\underline{1}$0$\underline{1}$

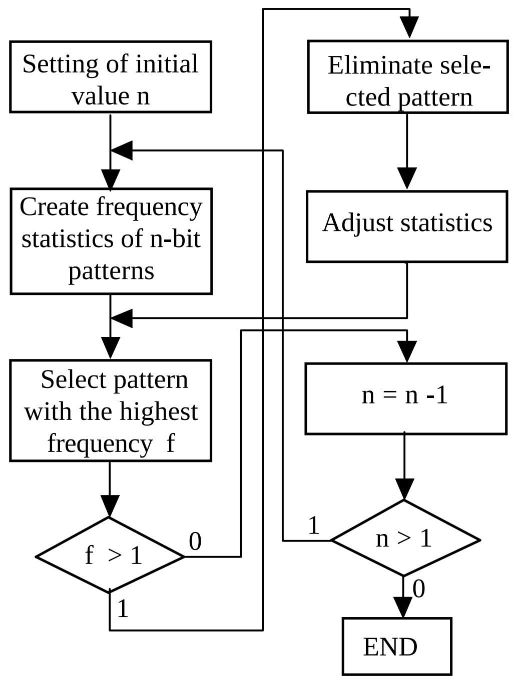

Setting of initial value n
Create frequency statistics of n-bit patterns
Select pattern with the highest frequency f
f > 1
0
1
Eliminate selected pattern
Adjust statistics
n = n -1
n > 1
1
0
END

Figure 4. Optimization algorithm block-diagram

IV.2 Pattern Selection

In the case of patterns with the same frequency, some decision criterion is necessary to make a selection. We have chosen a very simple one. Patterns with more bits must be implemented in adder/subtractor structures with a bigger wordlength, and thus result in a larger area. If two patterns have the same frequency, the smallest pattern is chosen for further processing. For example, pattern 101 is preferred over pattern 1001. In a case of two bit patterns of equal length, one additional criterion is added: the preference of adders over subtractors. This is because an adder structure is smaller than a subtractor of the same wordlength. For example, pattern 101 will be selected rather than $10\underline{1}$.

IV.3 Searching Algorithm

Because an exhaustive search is performed, the selection of an appropriate algorithm has crucial importance. We have decided to use a binary tree structure to hold the frequency statistics of patterns. As keys in the tree structure bit-patterns are used.
Initially, the pattern statistics are constructed as shown in figure 5. For each coefficient, a local tree is constructed.
After all patterns for that coefficients are evaluated, the global statistics tree holding information on the complete coefficient set, is updated.
After elimination of a pattern, the frequency of other patterns can also change, and therefore the frequency statistics must be re-evaluated after each elimination. Since the creation of a new global tree after each pattern elimination is unacceptable, an alternative method of adjusting the statistics is used (Figure 6). The idea is to use a difference frequency statistics for every changed coefficient that is processed. This local tree holds the information about changes in frequencies for the different subexpressions, after pattern elimination. The global tree can then be updated with this difference information after common subexpression removal. For example, in Figure 6, the frequency of pattern 101 frequency has changed from 2 to 1, due to elimination of the subexpression 1000001 out of the coefficient 0.10101010, so the difference frequency of pattern 101 is -1 (one less) thus the global frequency of this pattern must be also decremented. By backannotating the global tree only after common subexpression removal, the global statistics have to be created only once at the beginning.

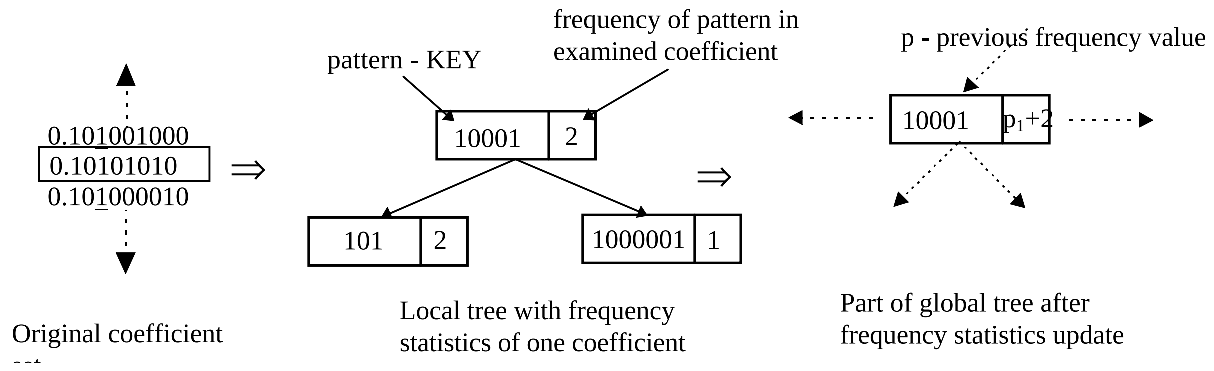

0.101001000
0.10101010
0.101000010
Original coefficient
⇒
pattern - KEY
frequency of pattern in examined coefficient
10001
2
101
2
1000001
1
Local tree with frequency statistics of one coefficient
⇒
p - previous frequency value
10001
p1+2
Part of global tree after frequency statistics update

Figure 5. Creation of a new frequency statictics

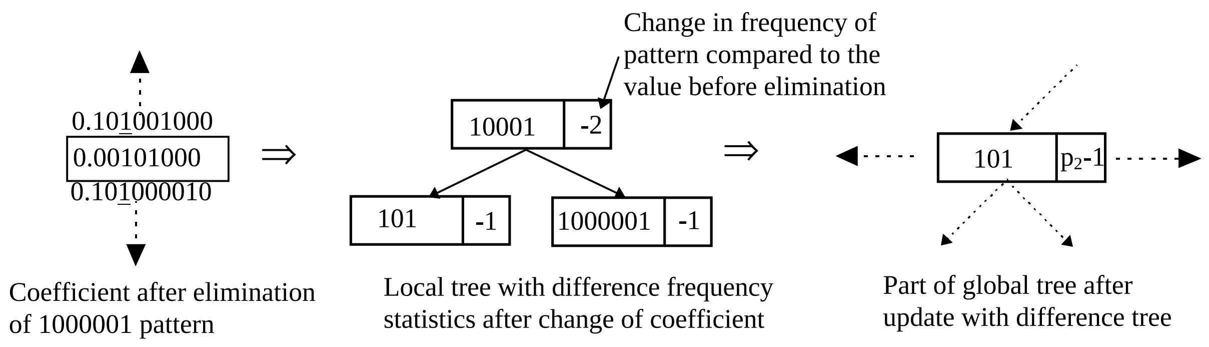

0.101001000
0.00101000
0.101000010
Coefficient after elimination of 1000001 pattern
Change in frequency of pattern compared to the value before elimination
10001
-2
101
-1
1000001
-1
Local tree with difference frequency statistics after change of coefficient
101
p2-1
Part of global tree after update with difference tree

Figure 6. Frequency statistics re-evaluation after 1000001 patern elimination

# of taps	8 bit CSD coeff.			12 bit CSD coeff.			16 bit CSD coeff.		
	init	opt	R	init	opt	R	init	opt	R
16	23	12	1.9	41	18	2.3	58	22	2.6
32	40	19	2.1	77	30	2.6	105	34	3.1
64	64	24	2.7	131	46	2.8	181	48	3.8
128	81	26	3.1	178	52	3.4	250	56	4.5
256	107	35	3.1	239	68	3.5	331	68	4.9

Table 3. Experimental results

V. Design Examples

In this section, we present the results that are obtained using the proposed algorithm. First, the performance of the approach is discussed, and next, a comparison with the work of other authors is made.

V.1 Performance

The CSD conversion as well as the CSE algorithm have been developed in C. In the first example, three sets of FIR filters with CSD coefficients of 8,12 and 16 bits chosen at random were processed. The number of taps was 16, 32, 64, 128 and 256 for each set. The results are given in Table 3 in terms of the number of adders-subtractors necessary to implement the structure of the multiplier block (See Figure 3.a). We give the number of add/sub before (init) and after optimization (opt) as well as the optimization ratio R=init/opt. The results demonstrate, that number of adders/subtractors for this random set of coefficients can be reduced by an average factor of 3.1 and the whole optimization works the better, the more coefficients are processed. Execution times of CSE algorithm for all examples were less than 1 second on a HP700 workstation.

In the second example, a Root Raised Cosine 64-tap FIR filter with impulse and frequency responses shown in Figures 7 and 8 respectively,10 bit CSD coefficients, 10 bit input and 16 bit output bus was optimized. After optimization, behavioral VHDL code was generated and synthesized using SYNOPSYS. Synthesis result are in Table 4. The area value is given as a sum of combinatorial and sequential area. Area and critical path (t_{crit}) values are evaluated for Alcatel-Mietec 0.7um 5V standard cell technology.

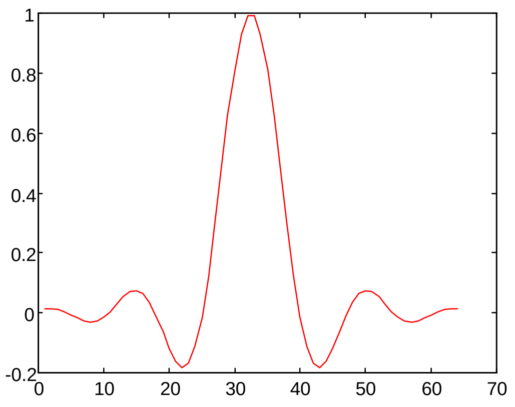


Figure 7. Filter impulse response

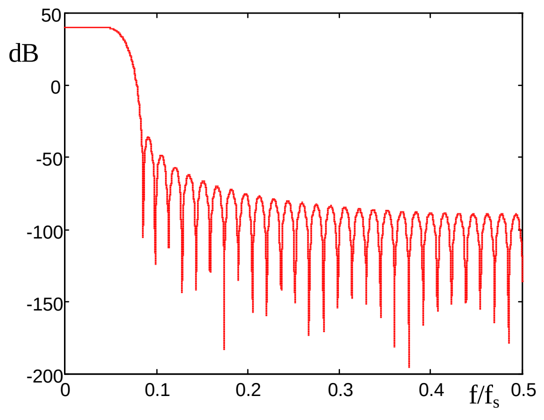


Figure 8. Filter frequency response

	Unoptimized	Optimized
Area-com.[mm^2]	5.181	4.4804
Area-seq [mm^2]	3.9415	3.9415
Area-tot. [mm^2]	9.1224	8.4219
t_{crit}[ns]	41.1	41.46

Table 4. Synthesis results

V.2 Related Work Comparison

Optimization of FIR filters by means of removal of redundant hardware was already reported by several authors [5 - 8]. A comparison was made based on results presented for random coefficients.
We compare the average optimization ratios R.
. Table 5 shows that the average optimization ratio of 3.1 that we obtain is clearly better than that what was presented before.

Author	**R**
Potkonjak [6]	1.4
Hartley [7]	2.1
Mehendale [5]	2.2
Demptser [8]	2.7
This work	3.1

Table 5. Related work results comparison

VI. Conclusion

In this paper, we presented an approach to the Multiple Constant Multiplication problem defined in [6] by exhaustive search. The results on our experimental filter set showed a significant reduction of hardware necessary in the multiplier block of a transposed form FIR filter with better performance than previously published papers. By including behavioral code generation as a postprocessing step of the optimization program, a tedious design subtask of broadband modem development has been automated and simplified.

Bibliography

[1] Digital Broadcasting Systems for Television, ETS 300 429, ETSI, 1994.

[2] Digital Audio Visual Council, “Lower Layer Protocols And Physical Interfaces”, DAVIC 1.0 Specification, Part 08, 1996.

[3] K. Hwang, “Computer Arithmetic”, Jhon Wiley, New York, 1979.

[4] H. Samueli, “The Design of Multiplierless Digital Data Transmission Filters with Powers-of-two Coefficients“, Proc. SBT/IEEE Int. Telecomm.Symp. September 1992, pp. 425 - 429

[5] Mehendale M., Sherlekar S.D., Venkatesh G., „Synthesis of Multiplier-less FIR Filters with Minimum Number of Additions“, IEEE Transactions on computer aided design 1995.

[6] Potkonjak M., Srivastava M.B., Chandrakasan P.A., „Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination“, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, Vol 15. No. 2. Feb. 1996

[7] Hartley R.I., „ Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers“, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 43. No 10. Oct. 1996

[8] A.G. Demptser, M.D. Macleod, “Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters“, IEEE Transactions on Circuits and Systems-II:Analog and Digital Signal Processing, Vol. 42, No 9, Sept. 1995.