(IJAER) 2015, Vol. No. 9, Issue No. VI, June **ISSN: 2231-5152**

OPTIMIZATION OF FIR FILTER USING MULTIPLE CONSTANT MULTIPLICATION

[1] S.Ateeb Ahmed, [2]Mr.S.Yuvaraj

[1]*Student, Department of Electronics and Communication/ VLSI Design*
SRM University, Chennai, India
[2]*Assistant Professor (O.G), Department of Electronics and Communication*
SRM University, Chennai, India

ABSTRACT

For implementing transposed direct-form FIR filters Multiple Constant Multiplications (MCM) are widely used. Constant multiplication methods are widely used for reducing computational complexity of implementation of FIR filters. In MCM the main focus has been on more on common sub expression elimination, the optimization of adder-trees, which sum up the computed sub-expressions for each coefficient, is largely omitted. In this paper it is identified that problem in the scheduling of adder-tree operations for the MCM block, and presented a mixed integer programming (MIP) based algorithm for more efficient MCM-based implementation of FIR filters.

INTRODUCTION

FINITE IMPLUSE RESPONSE (FIR) digital filters are widely used as a basic tool in many digital signal processing (DSP) and communication applications. The digital filters are frequently used in DSP by virtue of stability and easy implementation. The.complexity of a FIR filters are largely dominated by the multiplication of input sample with filter coefficients. But fortunately, the filter coefficients are constants for a given filter, so that multiplications are implemented by a network of adders, subtractors, and hardwired shifts, where the number of adders and subtractors are minimized by constant multiplication scheme. In case of a transposed direct-form FIR filter, the recent most input sample at any given clock period is multiplied with all the filter coefficients. A set of intermediate results are generated in this case, and shared across all the multiplications in order to minimize the total number of additions/subtractions using multiple constant multiplication (MCM) techniques.

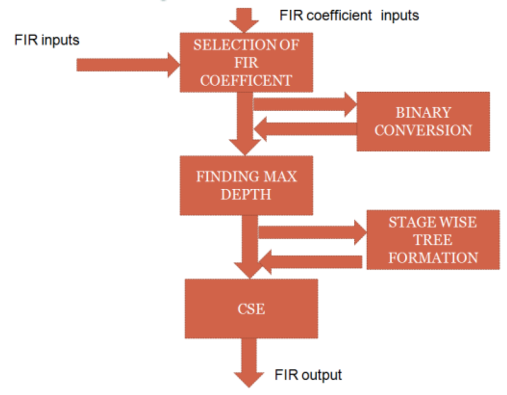


Fig. 1. Block diagram of MCM FIR filter

This paper presents a common sub expression elimination (CSE) method which used binary number representation as opposed to the commonly used canonical signed digit (CSD) representation. To reduce the complexity CSD representation can be used for the multiple coefficients. In signed digit representation a ternary representation with digits representing 1,0, and -1 is used where -1 represents subtraction. In this paper the idea is to show the efficiency of MCM compared to the conventional FIR filters and CSD filter. A great deal of research has been done to develop effective algorithms to identify the optimal set of non-redundant sub expressions to achieve the minimum number of logic operators and the minimum logic depth of the MCM . Irrespective of differences in methodology and the level of optimality, in all these works, after the common sub expression terms are determined and the ADD/SUB network of non-redundant sub expressions (or terms) is formed, the product value corresponding to each of the coefficients is computed by an adder-tree that sums up its relevant terms. As shown in Fig. 1b(b), two adder-trees are formed, for computing the product of a pair of coefficients using shifted versions of unique CS terms ‘1’, ‘10-1’ and ‘1001’ from the *term-networks* or *subexpression-networks*. Fig. 1b(a)). Each such intermediate result in an MCM process corresponds to one of the common sub-expressions (CS) of the set of constants to be multiplied.

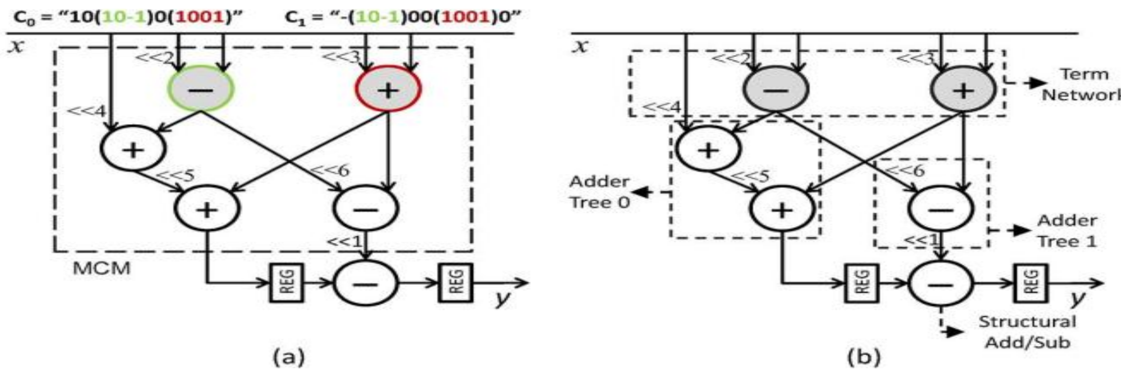


Fig. 1.(b) composition of the MCM block (a) MCM and common sub expression. (b) Term network and addrer tree for each coefficients.

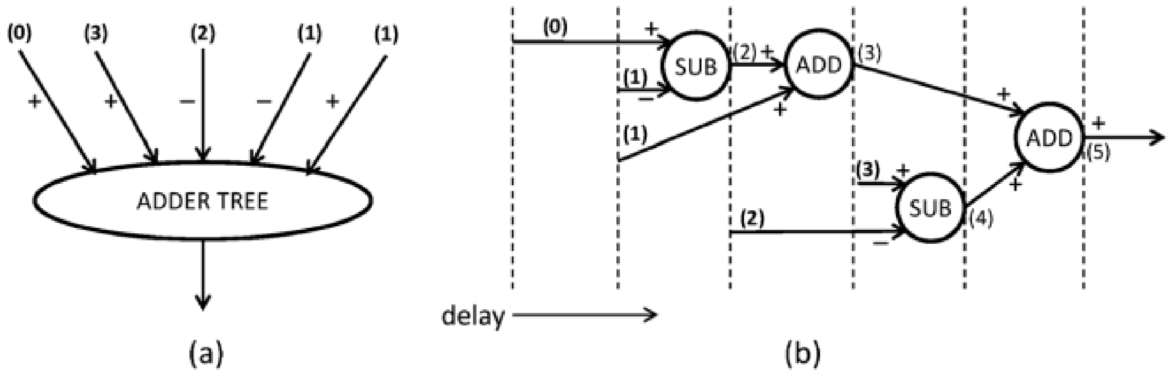


Fig. 2. (a) An example adder-tree with delays and signs on each input term, (b) An internal schedule with minimum delay.

ADDER-TREE SCHEDULING PROBLEM

Given input terms and their earliest $T=\{T_0,\ldots\ldots,T_{n-1}\}$ arrival time/delay d_i , the objective of an adder-tree scheduling algorithm is to define an assignment of binary addition and subtraction operations to sum up the input terms such that the total delay to produce the final output is minimized.

A *GREEDY ADDER TREE SCH*EDULING

The common practice of handling the summation of CS terms of each coefficient is to use the tree-height minimization algorithm to produce a height optimum adder-tree. The tree-height minimization algorithm iteratively collapses the Pair $\{T_i\ ,\ T_j\}$ with smallest delays using an

ADD/SUB to form a new term with delay $max\{D_i , D_j\}+1$, until a single term is reduced to. Fig. 2 gives an example of the schedule for an adder-tree on the left with minimum delay. Note that either a positive or negative sign is associated with each input term (see Fig. 2(a)), which denotes whether the corresponding term should be added to or subtracted from the summation. These signs also determine whether an addition operation or a subtraction operation should be used when the algorithm collapses a pair of terms in the adder-tree based on the following rules. (1) If two input edges are of the same sign, an ADD will be used; otherwise, it will be a SUB. (2) The sign of the output edge is always the same as that of the "left" input edge (i.e., the minuend edge in the subtraction case).Using these two rules, it is possible that the final term producing the summation result may carry a negative sign, such that a negation is needed after the adder-tree to correct the value. For an FIR filter, results from multiple adder-trees are accumulated by a structural adder-register line. So the negation can be eliminated by replacing the structural adder with a subtractor.

B. *COST MODEL*

In order to quantify and minimize the hardware cost of the adder-tree, we model the cost of ADD/SUB operations in this section based on the ripple carry implementation, which is most area efficient and will be picked up by the hardware compiler whenever the timing allows. Without loss of generality, for a single ADD/SUB operation, the pair of its input operands may be of different bit-widths, and one of them is to be left shifted by certain bit positions. We enumerate all the scenarios of shift-add/sub operations in Fig. 3.The cost calculation is done separately in three bit-segments. Starting from the least significant bit (LSB), the 1st segment covers the bit positions up to but not including the first bit of the shifted operand; the 3rd segment covers the bits corresponding to the sign extension bits of the sign extended operand; the 2nd segment takes the rest of the bit positions. Two cases of ADD operation are shown in Fig. 3(a) and (b). In both cases, the 2nd and 3rd segments are implemented by one Full Adder (FA) per bit, while the 1st segment cost nothing than wiring. Four cases of SUB operation are shown in Fig. 3(c)–(f). In the first two cases where the shift is with the minuend, the 1st segment is implemented by FAs with invertors on the minuend bits, except for a single special case at the LSB using direct wire connection1 to save a pair of FA and invertor. The 1st segments for the last two cases are wires. The 2nd segment for all cases is implemented in pairs of FAs and invertors. For the 3rd segment, when the sign extension bits are from the subtrahend, inverters are not needed since these bits simply take the value of the inverted sign of the subtrahend.

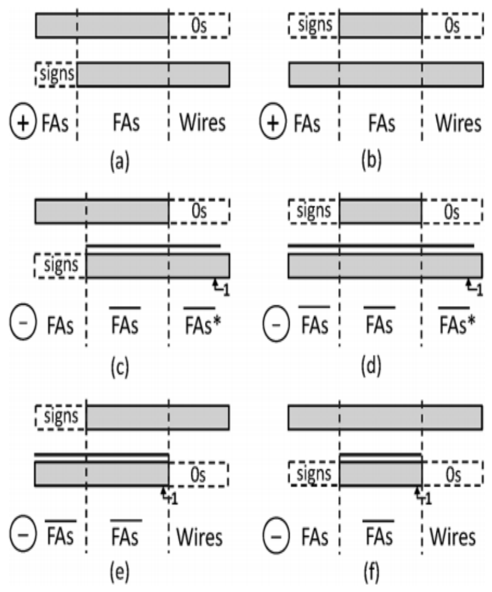


Fig. 3. Cost of ADD/SUB operation under various input scenarios. Notations: FA—Full Adder, aboveline—invertor (INV). (a)(b) Cases for ADD. (c)(d)(e)(f) Cases for SUB.

C. Logic Depth Relaxation

The clock performance of the entire FIR filter is decided by the largest of the delays of all coefficients. Assuming the delay of an ADD/SUB operator to be 1 unit, the delay of the constant multiplication by a coefficient can be simply measured by the number of ADD/SUB steps on a maximal path in the part of the network corresponding to the coefficient. We generally use logic depth to describe the required ADD/SUB steps. For a coefficient whose logic depth is less than the filter's logic depth, incrementing (relaxing) its logic depth may reduce the resource consumption.

Given an algorithm which computes the adder-tree of the minimum resource on a given depth for a coefficient, if is less than the filter's logic depth, one can always try increasing L by 1 and rescheduling onto L+1 a depth adder-tree for possible reduction of resource without degrading the filter's clock performance.

MINIMUM RESOURCE ADDER-TREE SCHEDULING USING MIP

Mixed integer programming (MIP) based formulation to schedule the adder-tree of an individual coefficient to minimize the hardware resource. As linearity is required in MIP, various techniques to transform modeling friendly non-linear expressions into linear equations and inequalities are indispensable and discussed in detail. This MIP procedure is then used in the next section as the building block for resource minimization for the entire FIR filter.

Given a set of input terms $T=\{T_0,\ldots\ldots,T_{n-1}\}$ and their earliest arrival time or delay D_i, we try to form an adder-tree of maximum depth to sum up the input terms and at the same time minimize the hardware resource used.

In order to model the above problem, we construct a complete binary tree $G($V, E) of depth L. Each node V_i € V on the tree is a position to hold a binary operator (ADD/SUB), and |V|= 2^l-1. Each edge E_i € E is a potential operand position to

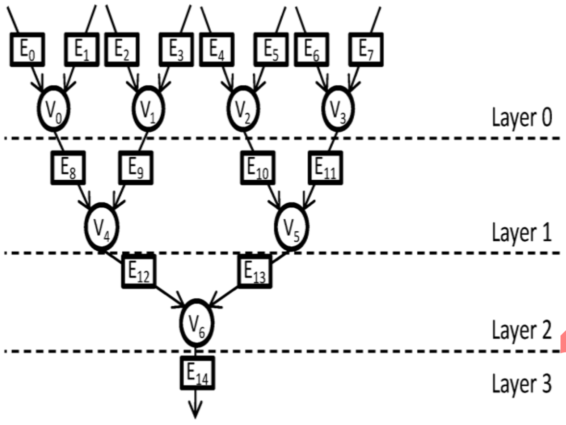


Fig. 5. Binary adder-tree of depth L=3 used for MIP modeling.

accommodate an input term. Potential operand position to accommodate an input term,and |E|= 2^{l+1}-1 . Fig. 5. shows an example of the decision tree when $L=3$. An input term T_i of delay D_i is schedulable on all edge positions of layer and downwards. For each input term T_i , we create a set of binary variable given as below equation is $TSet_i = \{t_{i,j} | \forall E_j \text{ of depths equal/greater than } D_i\}$. T_i is said to be scheduled on E_i if $t_{i,j}$ is assigned 1. The adder-tree scheduling problem is equivalent to finding an assignment of input terms to the edge positions on the binary tree such that the resource of the adder-tree is minimized. Constraints are formulated to ensure that the adder-tree summing up these input terms is correctly formed.

RESOURCE MINIMIZATION FOR THE ENTIRE FIR FILTER

In this section, we describe the top level algorithm to optimize the resource consumption of the entire FIR filter, based on the minimum resource of each individual coefficient returned by solving the MIP formulations discussed previously. The top level algorithm is based on the observation that logic depth relaxation on the coefficient may result in adder-tree of less resource. Without affecting the filter performance, logic depth relaxation is applicable to all the coefficients whose logic depths are smaller than the logic depth of the filter. We first compute the minimum logic depth required by a coefficient adder-tree, and then apply logic depth relaxation to improve the minimum resource whenapplicable.

A. Minimum logic depth of a coefficient Adder-Tree

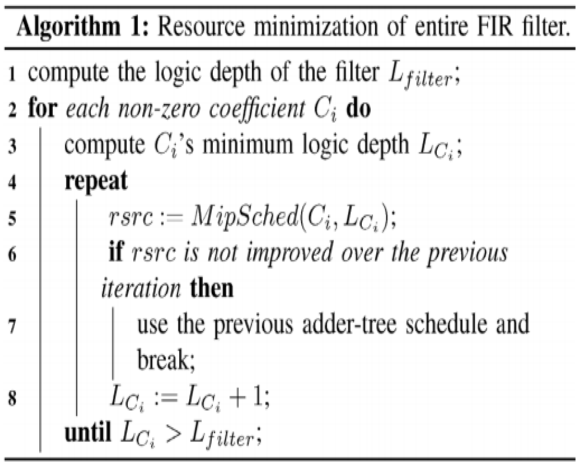
Firstly, the total number of ADD/SUB operator needed to sum up N input terms is N-1. Based on the binary tree model of the MIP formulation a input term T_i with latency /logic depth of D_i nullifies at least $2^{D}{}_{i}$-1 predecessor operator positions on the binary tree. As a result ,the minimum binary tree required to accommodating the input terms should consists of at least N-1+ $\sum_{i=0}^{N-1}(2^{Di}-1)$ operators positions. On the other hand binary tree of depths L contains 2^L-1operators.thus we have 2^L-1≥ N-1+ $\sum_{i=0}^{N-1}(2^{Di}-1)$ .The minimum depth required by the coefficient is computed by

$$L \geq [\log_2 N-1+\sum_{i=0}^{N-1} 2^{Di}]$$

The logic depth of the entire FIR filter takes the largest one among all its coefficients logic depths.

B. ***Resource minimization algorithm for FIR filter***

The algorithm for resource minimization of the entire FIR filter is elaborated in



Algorithm 1: Resource minimization of entire FIR filter.

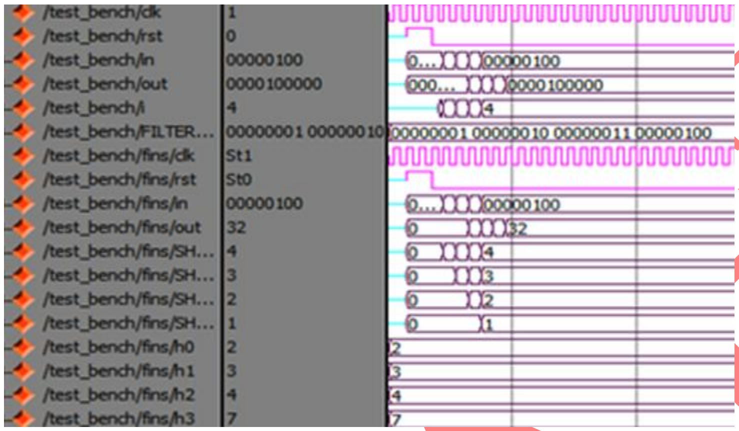
```
1 compute the logic depth of the filter $L_{filter}$;
2 for each non-zero coefficient $C_i$ do
3     compute $C_i$'s minimum logic depth $L_{C_i}$;
4     repeat
5         $rsrc := MipSched(C_i, L_{C_i})$;
6         if rsrc is not improved over the previous
          iteration then
7             use the previous adder-tree schedule and
              break;
8         $L_{C_i} := L_{C_i} + 1$;
      until $L_{C_i} > L_{filter}$;
```

Algorithm 1. For

particular coefficient, the MIP based resource minimization procedure is first applied to yield an adder-tree schedule at its minimum logic depth (line5). Further logic depth relaxation is attempted provided that the current depth does not exceed that of the filter's (line8), and the current attempt did improve the minimum resource (line 6). At the end of the algorithm, each coefficient is resource minimized with its adder-tree schedule. The total resource of the entire filter is minimized, without increasing its overall logic depth.

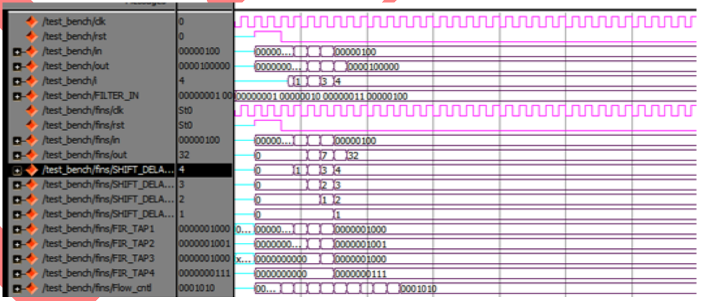
SIMULATION RESULTS

Multiple Constant Multiplication is implemented using shift add and subtract operations. Designed circuits are simulated using ModelSim ALTERA 6.4a. The normal conventional FIR Filter of four tap is shown in fig (5.1) and four tap filter using Canonical Sign Digit (CSD) is shown in fig (5.2).



```
/test_bench/clk            1
/test_bench/rst            0
/test_bench/in             00000100
/test_bench/out            0000100000
/test_bench/i              4
/test_bench/FILTER...      00000001 00000010
/test_bench/fins/clk       St1
/test_bench/fins/rst       St0
/test_bench/fins/in        00000100
/test_bench/fins/out       32
/test_bench/fins/SH...     4
/test_bench/fins/SH...     3
/test_bench/fins/SH...     2
/test_bench/fins/SH...     1
/test_bench/fins/h0        2
/test_bench/fins/h1        3
/test_bench/fins/h2        4
/test_bench/fins/h3        7
```

FIG 5.1: *Waveform of Conventional FIR Filter*



```
/test_bench/clk                  0
/test_bench/rst                  0
/test_bench/in                   00000100
/test_bench/out                  0000100000
/test_bench/i                    4
/test_bench/FILTER_IN            00000001 00
/test_bench/fins/clk             St0
/test_bench/fins/rst             St0
/test_bench/fins/in              00000100
/test_bench/fins/out             32
/test_bench/fins/SHIFT_DELA...   4
/test_bench/fins/SHIFT_DELA...   3
/test_bench/fins/SHIFT_DELA...   2
/test_bench/fins/SHIFT_DELA...   1
/test_bench/fins/FIR_TAP1        0000001000
/test_bench/fins/FIR_TAP2        0000001001
/test_bench/fins/FIR_TAP3        0000001000
/test_bench/fins/FIR_TAP4        0000000111
/test_bench/fins/Flow_cntl       0001010
```

FIG 5.2: Design of Filter using CSD

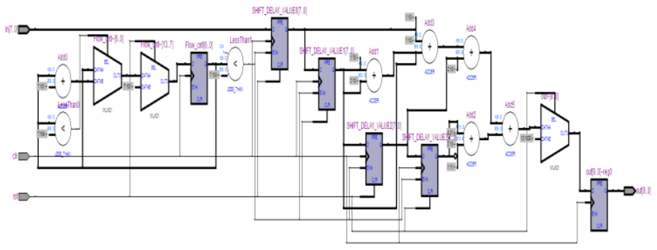


FIG 5.3: RTL schematic view of CSD filter

The performance analysis of both these filter are implemented in cadence synthesize tool where area and power are calculated . The below table illustrates the comparison differences of convention fir filter and csd filter.

TYPE	Area(logic cells)	Power(mW)	Frequency(MHz)
Multiplier Based	33396	0.669	286.2
CSD Based	18334	0.354	167.28

B) DESIGN OF FILTER USING CSD AND MIP (L=3)

For Binary Adder tree of the depth L=3 is used here since we are using the depth of three, we are going for the higher order filter. Figure (5.4) shows the waveform of Canonical Sign Digit (CSD) of ten tap fir filter and figure (5.5) shows the waveform result of Mixed Integer Programming (MIP) of ten tap fir filter for the logic depth of three.

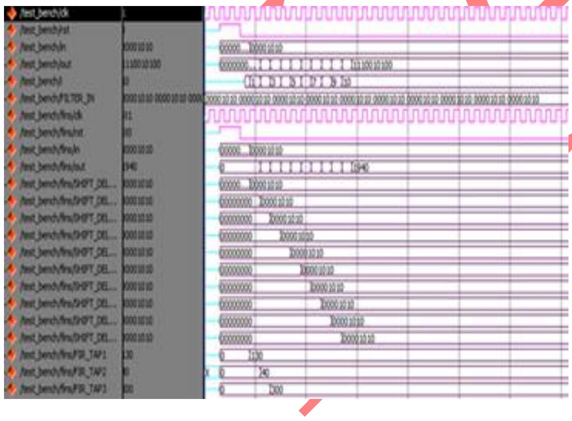


FIG 5.4: Waveform of filter CSD for L=3

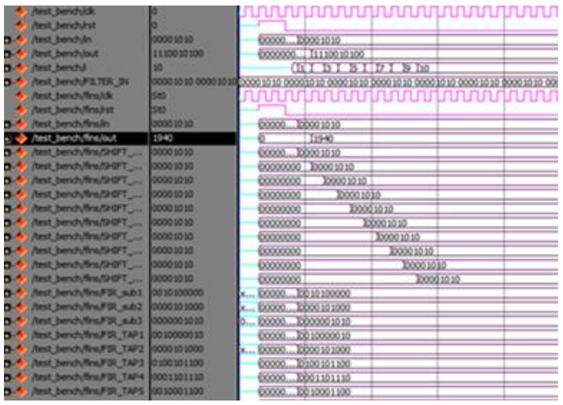


FIG 5.5: Waveform of filter MIP for L=3

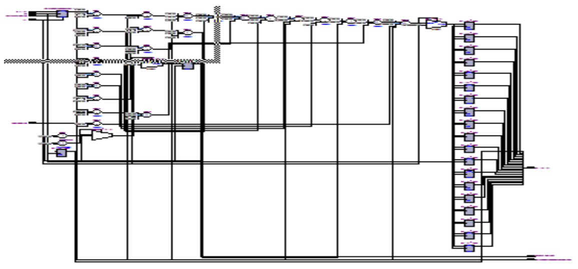


FIG 6.6: RTL schematic view of filter using MIP

TYPE	Area(logic cells)	Power(mW)	Frequency(MHz)
CSD Based	69575	1.346	68.68
MIP Based	57259	0.561	66.77

The table above shows the performance analysis of CSD and MIP based FIR filters.

CONCLUSION

In Optimization of Adder-Trees for Multiple Constant Multiplications for Efficient FIR Filter Implementation, we have identified the resource minimization problem in the scheduling of adder-tree operations for the MCM block of transposed direct-form FIR filter, and presented an MIP-based algorithm for exact bit-level resource optimization. Experimental result shows that up to 15% reduction of area and 11.6% reduction of power can be achieved on top of already optimized ADD/SUB networks of MCM blocks. Further exploration of efficient heuristic algorithms for resource minimization of adder-trees of FIR filters could be done in the future.

REFERENCES

[1] .M. B. Gately, M. B. Yeary, and C. Y. Tang, "*Multiple real-constant multiplication with improved cost model and greedy and optimal searches*," in Proc. IEEE ISCAS, May 2012, pp. 588–591.

[2] L. Aksoy, C. Lazzari, E. Costa, P. Flores, and J. Monteiro, "*Design of digit-serial FIR filters: Algorithms, architectures, and a CAD tool,*" IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 21, no. 3, pp. 498–511, Mar. 2013.

[3] Y. Voronenko and M. Püschel, "*Multiplierless multiple constant multiplication,*" ACM Trans. Algorithms, vol. 3, no. 2, 2007.

[4] L. Aksoy, C. Lazzari, E. Costa, P. Flores, and J. Monteiro, "*Design of digit-serial FIR filters: Algorithms, architectures, and a CAD tool,*" IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 21, no. 3, pp. 498–511, Mar. 2013.

[5] M. B. Gately, M. B. Yeary, and C. Y. Tang, "*Multiple real-constant multiplication with improved cost model and greedy and optimal searches,*" in Proc. IEEE ISCAS, May 2012, pp. 58.

[6] .M. Kumm, P. Zipf,M. Faust, and C.-H. Chang, "*Pipelined adder graph optimization for high speed multiple constant multiplication,*" in Proc IEEE ISCAS, May 2012, pp. 49–52.

[7] . R. Hartley and A. Casavant, "*Tree-height minimization in pipelined architectures*," in Proc. IEEE ICCAD, Nov. 1989.