INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 4, Issue 5 AUG 2015

Bit-Level Optimization of Adder-Trees for Multiple Constant Multiplications for Efficient FIR Filter Implementation

[1]**S. SRUTHI**, M.Tech, Vlsi System Design,
[2]**G. DEVENDAR** M.Tech, Asst.Professor, ECE Department,
[1,2]Tudi Ram Reddy Institute Of Technology And Sciences

***Abstract*:**In this paper, a set of operators suitable for digit-serial FIR filtering is presented. The canonical and inverted forms are studied. In each of these structures both the symmetrical and anti-symmetrical particular cases are also covered. In last two decades, many efficient algorithms and architectures have been introduced for the design of low complexity bit-parallel Multiple Constant Multiplications (MCM) operation which dominates the complexity of many digital signal processing systems. On the other hand, little attention has been given to the digit-serial MCM design that offers alternative low complexity MCM operations at the cost of an increased delay. In this topic, we address the problem of optimizing the gate-level area in digit -serial MCM designs and introduce high level synthesis algorithms, design architectures, and a computer aided design tool. The proposed optimization algorithms for the digit-serial MCM architectures in the design of digit-serial MCM operations and finite impulse response filters yields better performance compared with multiple Constant multipliers using Common Sub-expression Elimination (CSE) algorithm with high efficiency.

Keywords: 0–1 integer linear programming (ILP), digit-serial arithmetic, finite impulse response (FIR) filters, gate-level area optimization, multiple constant multiplications, Common Sub-expression Elimination (CSE) algorithm, Graph Base (GB) algorithm

I. INTRODUCTION

Finite Impulse Response (FIR) filters are of great importance in Digital Signal Processing (DSP) systems since their characteristics in linear-phase and feed-forward implementations make them very useful for building stable high-performance filters. The direct and transposed-form FIR filter implementations are illustrated in Fig. 1(a) and (b), respectively. Although both architectures have similar complexity in hardware, the transposed form is generally preferred because of its higher performance and power efficiency. The multiplier block of the digital FIR filter in its transposed form [Fig. 1(b)], where the multiplication of filter coefficients with the filter input is realized, has significant impact on the complexity and performance of the design because a large number of constant multiplications are required. This is generally known as the Multiple Constant Multiplications (MCM) operation and is also a central operation that has performance bottleneck in many other DSP systems such as fast Fourier Transforms, Discrete Cosine Transforms (DCTs), and error-correcting codes.

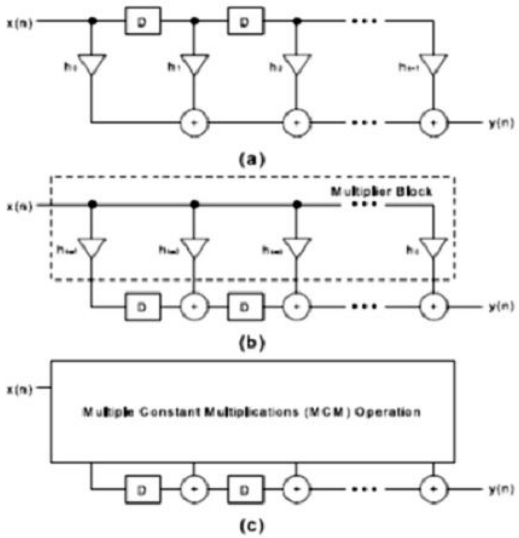

Fig.1: FIR filters implementations. (a) Direct form. (b) Transposed form with generic multipliers. (c) Transposed form with an MCM block.

In the last two decades, many efficient algorithms and architectures have been introduced for the design of low complexity bit-parallel multiple constant multiplications (MCM). Bit-parallel realization of the multiplication of a variable by a set of constants using only addition, subtraction, and shift operations has been explored extensively over the years as large number of constant multiplications dominates the complexity of many digital signal processing systems. On the other hand, digit -serial architectures offer alternative low-complexity designs since digit-serial operators occupy less area and are independent of the data word length. Also, it is mentioned that the full flexibility of a multiplier is not necessary for the constant multiplications, since filter coefficients are fixed and determined beforehand by the DSP algorithms. Hence, the multiplication of filter coefficients with the input data is generally implemented under a shift adds architecture, where each constant multiplication is realized using addition/subtraction and shift operations in an MCM operation.

One of the important implementation of the Multiplication of a variable with a set of constants also known as the MCM operation, is a central operation and achieves performance bottleneck in many DSP applications such as, error correcting codes, linear DSP transforms, and Finite Impulse Response (FIR) filters. In hardware, the multiplication operation is considered to be expensive, as it occupies significant area. Hence, constant multiplications are generally realized using only addition, subtraction, and shift operations.

Although area-, delay-, and power efficient multiplier architectures, such as Wallace and modified Booth multipliers, have been proposed, the full flexibility of a multiplier is not necessary for the constant multiplications, since filter coefficients are fixed and determined beforehand by the DSP algorithms. For the shift-adds implementation of constant multiplications, a straightforward method, generally known as digit based recoding, initially defines the constants in binary. Then, for each "1" in the binary representation of the constant, according to its bit position, it shifts the variable and adds up the shifted variables to obtain the result. As a simple example, consider the constant multiplications 29x and 43x.Their decompositions in binary are listed as follows:

$$29x = (11101)_{\text{bin}}x = x \ll 4 + x \ll 3 + x \ll 2 + x$$
$$43x = (101011)_{\text{bin}}x = x \ll 5 + x \ll 3 + x \ll 1 + x$$

Which requires six addition operations as illustrated in Fig. 2(a).

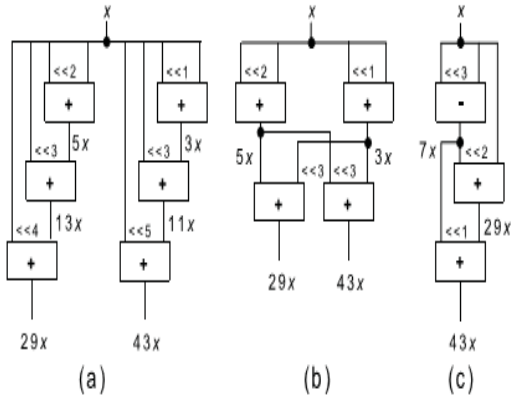

Fig. 2. Shift-adds implementations of 29xand 43x. (a) Without partial product sharing and with partial product sharing. (b) Exact CSE algorithm. (c) Exact GB algorithm

However, the digit-based recoding technique does not exploit the sharing of common partial products, which allows great reductions in the number of operations and, consequently, in area and power dissipation of the MCM design at the gate level. Hence, the fundamental optimization problem, called the MCM problem, is defined as finding the minimum number of addition and subtraction operations that implement the constant multiplications. Note that, in bit-parallel design of constant multiplications, shifts can be realized using only wires in hardware without representing any area cost.

In this paper, we initially determine the gate-level implementation costs of digit-serial addition, subtraction, and left shift operations used in the shift-adds design of digit-serial MCM operations. Then, we introduce the exact CSE algorithm that formalizes the gate-level area optimization problem as a 0–1 integer linear programming (ILP) problem when constants are defined under a particular number

representation. We also present a new optimization model that reduces the 0–1 ILP problem size significantly and, consequently, the runtime of a generic 0–1 ILP solver. Simulation results on a comprehensive set of instances show that the solutions of algorithms introduced in this paper lead to significant improvements in area of digit-serial MCM designs compared to those obtained using the algorithms designed for the MCM problem. Additionally, it is observed that the optimal tradeoff between area and delay in digit-serial FIR filter designs can be explored by changing the digit size d.

II. LITERATURE REVIEW

This section presents the main concepts related to the proposed algorithms, introduces the problem definitions, and gives an overview on previously proposed algorithms.

A. Number Representation

The binary representation decomposes a number in a set of additions of powers of 2. The representation of numbers using a signed digit system makes use of positive and negative digits, {1, 0, −1}. The CSD representation is a signed digit system that has a unique representation for each number and verifies the following main properties: 1) two nonzero digits are not adjacent; and 2) the number of nonzero digits is minimum. Any n digit number in CSD has at most [(n+1)/2] nonzero digits and, on average, the number of non zero digits is reduced by 33% when compared to binary

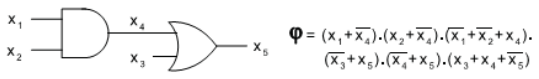

Fig.3.Combinational circuit and its corresponding CNF formula

B. Boolean Satisfiability

A Boolean function : {0, 1}n→ {0, 1} can be denoted by a propositional formula. The conjunctive normal form (CNF) is a representation of a propositional formula consisting of a conjunction of propositional clauses where each clause is a disjunction of literals and a literal l j is either a variable X_j or its complement $\sim X_j$. Note that, if a literal of a clause assumes value 1, then the clause is satisfied. If all literals of a clause assume the value 0, then the clause is unsatisfied. The Satisfiability (SAT) problem is to find an assignment on n variables of the Boolean formula in CNF that evaluates the formula to 1, or to prove that the formula is equal to the constant 0.

The CNF formula of a combinational circuit is the conjunction of the CNF formulas of each gate, where the CNF formula of each gate denotes the valid input–output assignments to the gate. The derivation of CNF formulas of basic logic gates can be found in [19]. In this Boolean formula, the first three clauses represent the CNF formula of a two-input AND gate, and the last three clauses denote the CNF formula of a two input OR gate. Observe from Fig. 3 that the assignment x1 = x3 = x4 = x5 = 0 and x2 = 1 makes the formula ϕ equal to 1, indicating a valid assignment. However, the assignment x1= x3 = x4 = 0 and x2 = x5 = 1 makes the last clause of the formula equal to 0 and, consequently, the formula ϕ, indicating a conflict between the values of the inputs and output of the OR gate.

C. 0–1 ILP

The 0–1 ILP problem is the minimization or the maximization of a linear cost function subject to a set of linear In (1), in w is an integer value associated with each of n variables x j, $1 \leq j \leq n$, in the cost function, and in (2), $A \cdot x \geq b$ denotes the set of m linear constraints, where $b \in Z^M$ and $A \in Z^m \times Z^n$

.

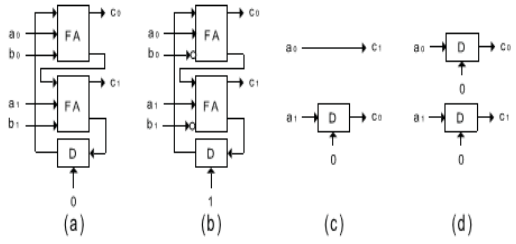

Fig. 4. Digit-serial operations when d is equal to 2. (a) Addition operation. (b)Subtraction

operation. (c) Left shift by one time. (d) Left shift by two times

D. Digit-Serial Arithmetic

In digit-serial designs, the input data is divided into d bits and processed serially by applying each d-bit data in parallel. The special cases, called bit-serial and bit-parallel, occur when the digit size d is equal to 1 and equal to input data word length, respectively. The digit-serial computation plays a key role when the bit-serial implementations cannot meet the delay requirements and the bit parallel designs require excessive hardware. The digit-serial addition, subtraction, and left shift operations are depicted in Fig. 4 using a digit size d equal to 2, where the bits with index 0 and 1 denote the least significant bit (LSB) and the most significant bit (MSB), respectively.

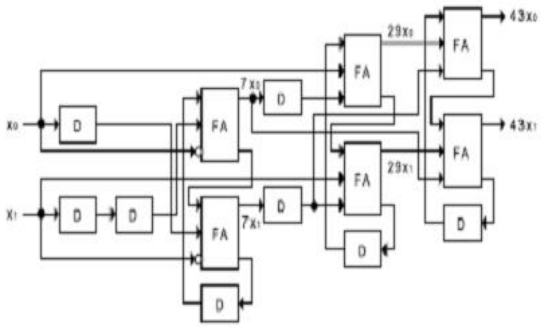

Fig 5: Digit-serial design of shift-adds implementation of 29x and 43x given in Fig. 2(c) when d is 2.

As an example of digit-serial realization of constant multiplications under the shift-adds architecture, Fig. 5 presents the digit-serial implementation of 29x and 43x illustrated in Fig. 2(c) when d is 2. For the sake of clarity, the initializations of D flip-flops are omitted in this figure. As can be easily observed, the network includes two digit-serial additions, one digit-serial subtraction, and five D flip-flops for all the left shift operations. In this network, at each clock cycle, two bits of the input data x (x1x0) are applied to the network input, and at the outputs of each digit-serial addition/subtraction operation two bits of a constant multiplication are computed. In general, d bits are processed at each clock cycle.

The digit-serial design of the MCM operation occupies significantly less area when compared to its bit-parallel design since the area of the digit-serial design is not dependent on the bit width of the input data. However, the latency is determined in terms of clock cycles as

$$L_{MCM} = \left\lceil \frac{(bw + N)}{d} \right\rceil \quad (5)$$

Where N is the bit width of the input variable x, bw is the maximum bit width of the constants to be implemented, and d is less than N. As a sign extension, d × LMCM − N bits must be padded to the input data x, which are zeros if x is an unsigned input, or sign bits otherwise.

E. Problem Definitions

For the multiplier less realization of constant multiplications, the fundamental operation, called A-operation in [11], is an operation with two integer inputs and one integer output that performs a single addition or a subtraction, and an arbitrary number of shifts. It is defined as follows:

$$w = A(u, v) = |2^{l_1}u + (-1)^s 2^{l_2}|2^{-r} \quad (6)$$

where $s \in \{0, 1\}$ is the sign, which determines if an addition or a subtraction operation is to be performed, $l1, l2 \geq 0$ are integers denoting left shifts of the operands, and $r \geq 0$ is an integer indicating a right shift of the result. The MCM problem [11] can be defined as follows

Definition 1 (The MCM Problem): Given the target set composed of positive and odd unrepeated target constants to be implemented, $T = \{t1, \ldots, tn\} \subset N$, find the smallest ready set, $R = \{r0, r1, \ldots, rm\}$, with $T \subset R$, under the conditions of $r0 = 1$ and for all rk with $1 \leq k \leq m$, there exist ri , $r j$ with $0 \leq i, j < k$ and an A-operation $rk = A(ri , r j)$.Hence, the number of operations required to be implemented for the MCM problem is $|R| - 1$ as given in [11]. Note that the MCM problem is an NP-complete problem Contrary to the bit-parallel MCM design, as described in Section II-D, shifts require D flip-flops in a digit-serial MCM design. Hence, the problem of optimizing the

number of addition, subtraction, and shift operations is defined as follows.

Definition 2 (The MCM-DS Problem): Given the target set $T = \{t1, \ldots, tn\} \subset N$, find the ready set $R = \{r0, r1, \ldots, rm\}$ such that under the same conditions on the ready set given in Definition 1, the set of A-operations include the minimum number of addition, subtraction, and shift operations. In the digit-serial architecture, it is assumed that an A-operation that generates a constant multiplication has always a right shift equal to zero due to the excessive hardware required for the control logic. Note that a constant multiplication is rarely realized by such an A-operation in GB algorithms and it never occurs in CSE algorithms. The optimization of gate-level area problem in digit-serial MCM operation can be defined as follows.

III EXACT CSE ALGORITHM

The exact CSE algorithm consists of four main steps. First, all possible implementations of constants are extracted from the nonzero digits of the constants defined under a number representation: binary, CSD, or MSD. Then, the implementations of constants are represented in terms of a Boolean network. Third, the gate-level area optimization problem is formalized as a 0–1 ILP problem with a cost function to be minimized and a set of constraints to be satisfied. Finally, a set of operations that yields the minimum area solution is obtained using a generic 0–1 ILP solver. These four steps are described in detail next.

A. Finding the Partial Terms

In the preprocessing phase, the constants to be multiplied by a variable are converted to positive, and then made odd by successive divisions by 2. The resulting constants are stored without repetition in the target set T. Thus, T includes the minimum number of necessary constants to be implemented. The part of the algorithm where the implementations of the target constants and partial terms are found is as follows.

1)Take an element from T, ti ,find its representation(s) under the given number representation, and store it(them) in a set called S. Form an empty set Oi associated with ti ,that will include the inputs and the amount of left shifts at the inputs of all addition/subtraction operations which generate ti .

2) For each representation of ti in the set S.

a) Compute all non symmetric partial term pairs that cover the representation of ti.

b) In each pair, make each partial term positive and odd, and determine its amount of left shift.

c) Add each pair to the set Oi with the amount of left shifts of partial terms.

d) Add each partial term to T, if it does not represent the input that the constants are multiplied with, i.e., denoted by 1, and is not in T

3) Repeat Step 1 until all elements of T are considered. Observe that the target set T only includes the target constants to be implemented in the beginning of the iterative loop, and in later iterations it is augmented with the partial terms that are required for the implementation of target constants. All possible implementations of an element in the target set ti are found by decomposing the nonzero digits in the representation of, ti, into two partial terms. As an example, consider 25 as a target constant defined under MSD, which has two representations 011001 and 10 001. All possible implementations of 25 are given in Fig. 6.

$$25 = \begin{cases} 011001 = \begin{cases} 010000 + 001001 = 1 \ll 4 + 9 \\ 001000 + 010001 = 1 \ll 3 + 17 \\ 000001 + 011000 = 1 + 3 \ll 3 \end{cases} \\ 10\bar{1}001 = \begin{cases} 100000 + 00\bar{1}001 = 1 \ll 5 - 7 \\ 00\bar{1}000 + 100001 = -1 \ll 3 + 33 \\ 000001 + 10\bar{1}000 = 1 + 3 \ll 3 \end{cases} \end{cases}$$

Fig 6: Possible implementations of 25 under MSD representation. includes only AND & OR gates. Its properties are given as follows.

B. Construction of the Boolean Network

After all possible implementations of target constants and partial terms are found, they are represented in a network that includes only AND and OR gates. Its properties are given as follows.

1) The primary input of the network is the input variable to be multiplied with the constants.
2) An AND gate in the network represents an addition/subtraction operation and has two inputs.
3) An OR gate in the network represents a target constant or a partial term and combines all its possible implementations.
4) The outputs of the network are the OR gate outputs associated with the target constants.

The Boolean network is constructed as follows.
1) Take an element from T, ti.
2) For each pair in Oi, generate a two-input AND gate. The inputs of the AND gate are the elements of the pair, i.e., 1, denoting the input that the constants are multiplied with, or the outputs ofORgates representing target constants or partial terms in the network.
3) Generate an OR gate associated with ti, where its inputs are the outputs of the AND gates determined in Step 2.
4) If ti is a target constant, make the output of the corresponding OR gate an output of the network.
5) Repeat Step 1 until all elements inTare considered.

The network generated for the target constant 25 defined under MSD is given in Fig. 7, where one-input OR gates for the partial terms 7, 9, 17, and 33 are omitted and the type of each operation is shown inside of each AND gate.

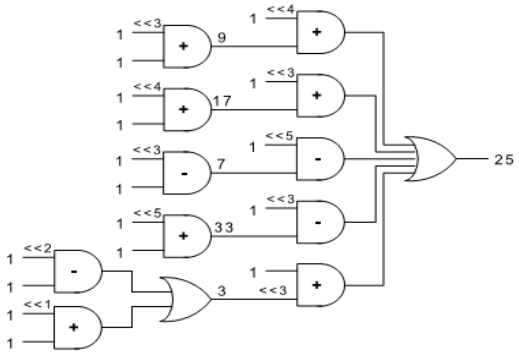

Fig.7. Network constructed for the target constant 25 under MSD

C. Formalization of 0-1 ILP Problem

We need to include optimization variables into the network, so that we can easily formalize the gate-level area optimization problem as a 0–1 ILP problem. The optimization variables are associated with two parameters that have different implementation costs at the gate level, i.e., addition/subtraction operations and left shifts of constants (partial terms and target constants).

For each AND gate that represents an addition/subtraction operation in the network, we introduce an optimization variable associated with the operation, i.e., opt a±b, where a and b denote the inputs of an operation and we add this variable to the input of the AND gate. The inclusion of these optimization variables into the network can be done in two ways.
Model 1: For each AND gate in the network representing an addition/subtraction operation, if an input signal ins is shifted by ls > 0 times, then we include ls additional inputs standing for the optimization variables associated with the ls left shift of the input signal ins, i.e.,

$$opt_{ins \ll 1},\ opt_{ins \ll 2},\ \ldots,\ opt_{ins \ll ls}.$$

Model 2: Initially, for each constant c with mlsc greater than zero, we generate a chain of mlsc − 1 AND gates with two inputs, where the inputs of the first AND gate of the chain are optc_1 and optc_2, and the inputs of the ith AND gate are optc_i+1 and the output of the (i − 1)th (previous) AND gate in the chain, where $2 \leq i \leq mlsc - 1$. Then, for each AND gate representing an addition/subtraction operation, if an input signal ins is shifted by ls > 0 times, we add a single input to the AND gate. This input is if ls is equal to 1, or otherwise, the output of the (ls − 1)th AND gate in the chain of AND gates including the optimization variables for the mlsins left shift of the input signal ins.

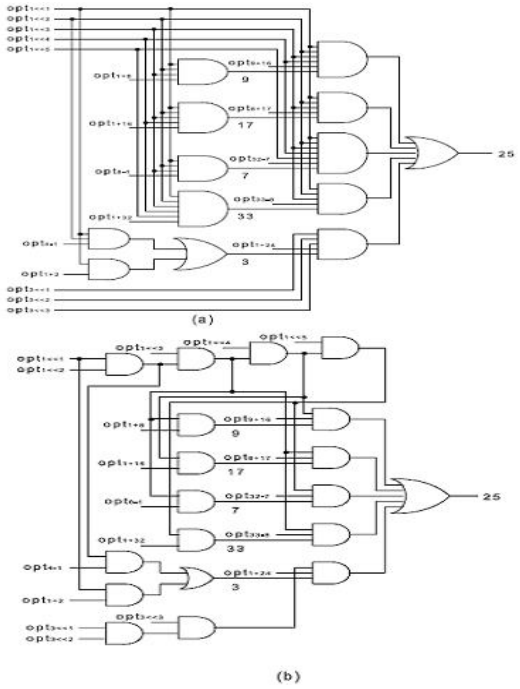

Fig 8: Networks constructed for the target constant 25 under MSD after the optimization variables are added. (a) Under Model 1. (b) Under Model 2.

Some simplifications in the network can be also achieved. The input variable x denoted by 1 can be eliminated from the inputs of the AND gates, because its logic value is always 1 (i.e., it is always available). Figs. 8(a) and (b) illustrate the networks generated for the target constant 25 under MSD after the simplifications are done and the optimization variables are added under Models 1 and 2, respectively. After the optimization variables are added into the network, the 0–1 ILP problem is generated. The cost function of the 0–1 ILP problem is constructed as the linear function of optimization variables, where the cost value of each optimization variable is determined as described previously.

D. Finding the Minimum Area Solution

A generic 0–1 ILP solver will search for the minimum value of the cost function on the generated 0–1 ILP problem by satisfying the constraints that represent how target constants and partial terms are implemented. The set of operations that yields the minimum area solution consists of the addition/subtraction operations whose optimization variables are set to 1 in the solution obtained by the 0–1 ILP solver.

IV. APPROXIMATE GB ALGORITHM

Note that the solution of an exact CSE algorithm described in privies Section is not the global minimum since all possible implementations of a constant are found from its representation. Also, the optimization of gate -level area problem in digit-serial MCM design is an NP-complete problem due to the NP - completeness of the MCM problem. Thus, naturally, there will be always 0 –1 ILP problems generated by the exact CSE algorithm that current 0–1 ILP solvers find difficult to handle. Hence, the GB heuristic algorithms, which obtain a good solution using less computational resources, are indispensable. In our approximate algorithm called MINAS-DS, as done in algorithms designed for the MCM problem given in Definition1, we find the fewest number of intermediate constants such that all the target and intermediate constants are synthesized using a single operation. However, while selecting an intermediate constant for the implementation of the not -yet synthesized target constants in each iteration, we favor the one among the possible intermediate constants that can be synthesized using the least hardware and will enable us to implement the not -yet synthesized target constants in a smaller area with the available constants. After the set of target and intermediate constants that realizes the MCM operation is found, each constant is synthesized using an A-operation that yields the minimum area in the digit -serial MCM design. In MINAS-DS, the area of the digit-serial MCM operation is determined as the total gate -level implementation cost of each digit-serial addition, subtraction, and shift operation under the digit size parameter d.

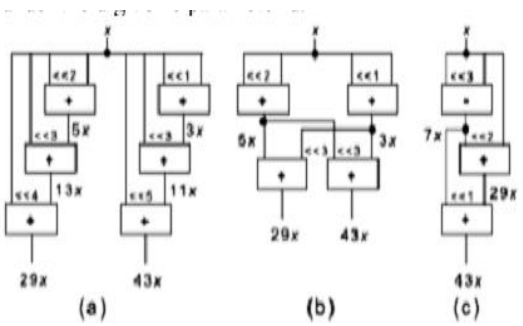

Fig. 9. Shift-adds implementations of 29 x and 43x. (a) Without partial product sharing and with partial product sharing. Exact CSE algorithm. (c) Exact GB algorithm

V. COMPUTER- AIDED DESIGN TOOL

This section initially presents the design of a digit-serial MCM operation under the shift adds architecture. Then, a generic digit-serial constant multiplier architecture adapted from ,which is used for an alternative digit serial realization of the MCM block and for comparison with the shift-adds architecture.

A. Design of Digit-Serial MCM Operations under the Shift-adds Architecture

In this case, we use the solution of a high-level algorithm on an MCM instance that consists of A-operations implementing the constant multiplications. Initially, to design the necessary circuit for the implementation of left shift operations, for each constant c in the solution of a high-level algorithm we find the maximum amount of left shift that c has, i.e. mlsc.

B. Design of Digit-Serial MCM Operations Using Digit-Serial Constant Multipliers

Generic digit-serial multiplier architectures in which both operands are time-variant. However, these architectures are not flexible enough to take the advantage of constant multiplications. On the other hand, bit serial constant multiplier architectures in which one operand is constant (time-invariant).In these constant multiplier architectures, the hardware of the design is significantly reduced with respect to the generic digit-serial multipliers by utilizing the nonzero digits of the constant to be multiplied by the input variable x. Moreover CSE technique used to maximize the sharing of partial products among the constant multiplications was also proposed.

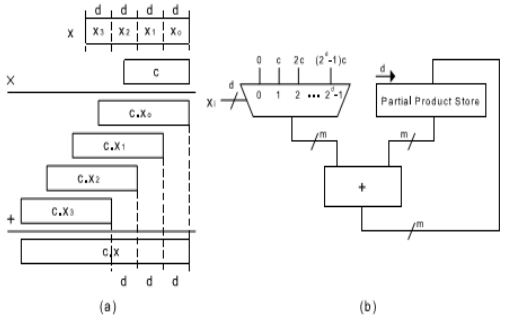

Fig. 10. Digit-serial constant multiplier using the sequential multiplication algorithm. (a) Illustrative example. (b) Design architecture.

VI. SIMULATION RESULTS

The proposed design has been simulated using Xilinx and Modelsim, the wave form obtained after simulating is as shown in fig .11,12,13.

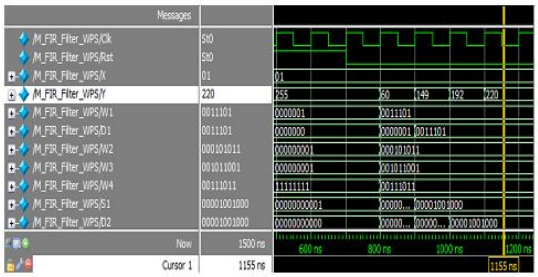

Fig 11 Output for FIR filter with digit based recoding algorithm

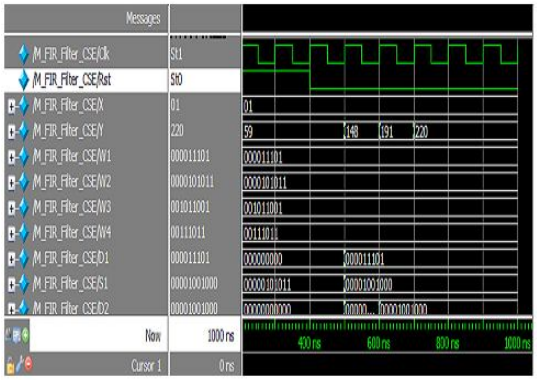

Fig 12 Output for FIR filter with CSE algorithm

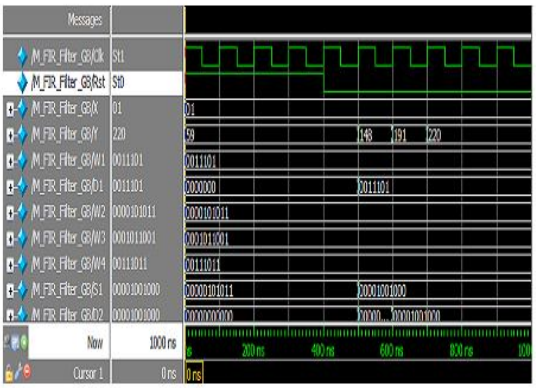

Fig 13 Output for FIR filter with GB algorithm

The RTL schematic of the proposed system is as shown in fig. 14, then it simulated result is implemented on FPGA Spartan 3E.

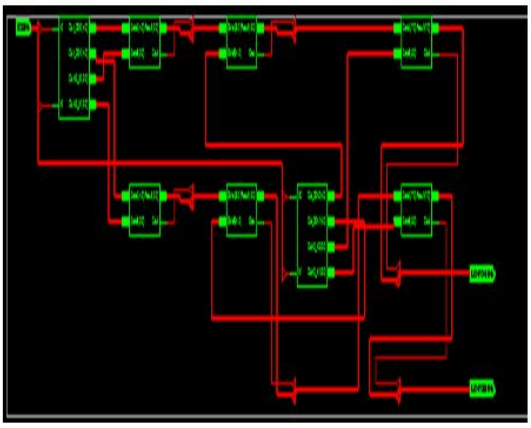

Fig 14 RTL schematic view of FIR filter

VII. CONCLUSION

In this paper, we introduced the 0–1 ILP formalization for designing digit-serial MCM operation with optimal area at the gate level by considering the implementation costs of digit-serial addition, subtraction, and shift operations. Since there are still instances with which the exact CSE algorithm cannot cope, we also proposed an approximate GB algorithm that finds the best partial products in each iteration which yield the optimal gate level area in digit-serial MCM design. This paper also introduced the design architectures for the digit-serial MCM operation and a CAD tool for the realization of digit-serial MCM operations and FIR filters. The proposed results indicate that the complexity of digit serial MCM designs can be further reduced using the high-level optimization algorithms proposed in this paper. It was shown that the realization of digit-serial FIR filters under the shift-adds architecture yields significant area reduction when compared to the filter designs whose multiplier blocks are implemented using digit-serial constant multipliers.

REFERENCES

[1] L. Wanhammar, DSP Integrated Circuits. New York: Academic, 1999.
[2] C. Wallace, “A suggestion for a fast multiplier,” IEEE Trans. Electron. Comput., vol. 13, no. 1, pp. 14–17, Feb. 1964.
[3] W. Gallagher and E. Swartzlander, “High radix booth multipliers using reduced area adder trees,” in Proc. Asilomar Conf. Signals, Syst. Comput., vol. 1. Pacific Grove, CA, Oct.–Nov. 1994, pp. 545–549.
[4] J. McClellan, T. Parks, and L. Rabiner, “A computer program for designing optimum FIR linear phase digital filters,”IEEE Trans. AudioElectroacoust., vol. 21, no. 6, pp. 506–526, Dec. 1973.
[5] H. Nguyen and A. Chatterjee, “Number-splitting with shift-and-add decomposition for power and hardware optimization in linear DSP synthesis,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,vol.8, no. 4, pp. 419–424, Aug. 2000.
[6] M. Ercegovac and T. Lang,Digital Arithmetic. San Mateo, CA: Morgan Kaufmann, 2003.
[7] R. Hartley, “Subexpression sharing in filters using canonic signed digit multipliers,”IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 43, no. 10, pp. 677–688, Oct. 1996.
[8] I.-C. Park and H.-J. Kang, “Digital filter synthesis based on minimal signed digit representation,” inProc. DAC, 2001, pp. 468–473.
[9] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, “Exact and approximate algorithms for the optimization of area and delay in multiple constant multiplications,”IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 27, no. 6, pp. 1013–1026, Jun. 2008.

[10] A. Dempster and M. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 42, no. 9, pp. 569–577, Sep. 1995.
[11] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication,"ACM Trans. Algor., vol. 3, no. 2, pp. 1–39, May 2007.
[12] L. Aksoy, E. Gunes, and P. Flores, "Search algorithms for the multiple constant multiplications problem: Exact and approximate," J. Microprocess. Microsyst., vol. 34, no. 5, pp. 151–162, Aug. 2010.
[13] R. Hartley and K. Parhi, Digit-Serial Computation. Norwell, MA: Kluwer, 1995.