International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 10, October - 2013

A New Design and an Architecture for FIR filter with Flexibility and Power Reduction

Sruthy K
ME VLSI
KALAIGNAR KARUNANIDHI INSTITUTE OF TECHNOLOGY
Coimbatore
India

Mr.S. Maria Antony
Asst.Professor
KALAIGNAR KARUNANIDHI INSTITUTE OF TECHNOLOGY
Coimbatore
India

Ms. S.Kavitha
ME VLSI
KALAIGNAR KARUNANIDHI INSTITUTE OF TECHNOLOGY
Coimbatore
India

Abstract

Efficient algorithms and architectures have been introduced for the design of low complexity bit-parallel multiple constant multiplications (MCM)[1]. But all these results in an additional complexity for the system operation, digit-serial MCM design[1] offers alternative low complexity MCM operations at the cost of an increased delay. Canonical signed digit based recoding is proposed with a new reconfigurable architecture[21] of low complexity.

A better digit-based method decomposes into both adds and subtracts by recoding the number into the canonical signed digit (CSD) representation [Avizienis 1961], which allows negative digits. The straightforward method for decomposing the multiplication into adds and

shifts translates 1's in the binary representation of the constant 't' into shifts, and adds up the shifted inputs. For example, consider t = 71,

$$71x = 1000111_2 x = x \ll 6 + x \ll 2 + x \ll 1 + x,$$

with 3 shift operations and 3 addition operations. The proposed method is using CSD[11], the previous example can be improved to use only 2 add/subtract operations viz:

$$1000111_2 x = 100100\overline{1}_{\mathrm{CSD}} x = x \ll 6 + x \ll 3 - x$$

The proposed architecture is capable of operating for different word length filter coefficients without any overhead in the hardware circuitry. Dynamically reconfigurable filters[21] can be efficiently implemented by using CSD[11]. Design examples show that the proposed architectures offer good area and power reductions and speed improvement compared to the best existing one. Algorithms and schemes for computing several common arithmetic expressions defined in the complex domain as hardware-implemented operators. Mapping the expression to a system of linear equations, apply a complex-to-real transform, and compute the solutions to the linear system using a digit-by-digit, the

most significant digit first, recurrence method are also tested.

Index Terms— Complex arithmetic, Canonical signed digit(CSD),high level synthesis, reconfigurable, Finite Impulse Response (FIR) filters .

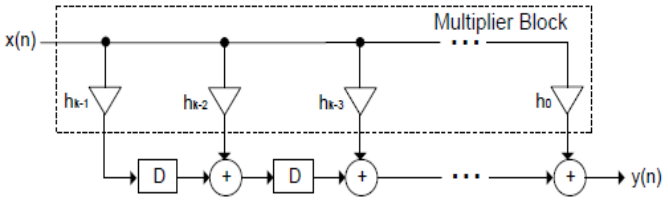

Figure 1: Transposed form of a digital FIR filter design

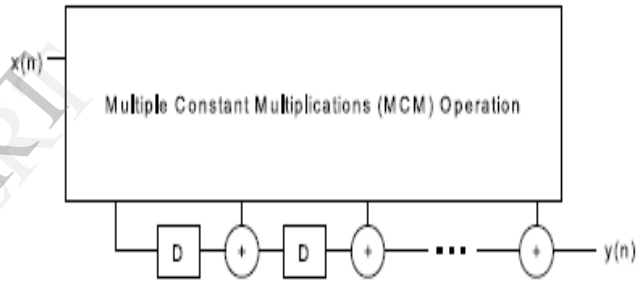

Figure 2: Transposed form mcm block.

I. INTRODUCTION

In digital signal processing (DSP) systems finite impulse response (FIR) filters are of great importance. Transposed Direct form is the usually used computational technique as it offers a higher power efficiency and hence higher performance. The multiplier block as of in fig – b involves multiplication of a single input with different filter coefficients which are also constants. Full flexibility of a multiplier is not necessary for the constant multiplications. Here only constants are to be multiplied and no floating point multiplication is involved.

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 .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[1].

The complexity of FIR filters is dominated by the complexity of coefficient multipliers. It is well known that the common sub expression elimination (CSE) methods based on canonical signed digit (CSD)[11] coefficients produce low complexity FIR filter coefficient multipliers. The goal of CSE[1] is to identify multiple occurrences of identical bit patterns that are present in the CSD [11]representation of coefficients, and eliminate these redundant multiplications. A modification of the 2-bit CSE[1] technique for identifying the proper patterns for elimination of redundant computations and to maximize the optimization impact was proposed. Another technique was to minimize the logic depth (LD) (LD is defined as the number of adder-steps in a maximal path of decomposed multiplications and thus to improve the speed of operation. The proposed method use binary common sub expression elimination (BCSE)[21] method which provides improved adder reductions and thus low complexity FIR filters . In a method based on the pseudo floating point method was used to encode the filter coefficients and thus to reduce the complexity of the filter. But the method is limited to filter lengths less than 40.In general, the methods are only suitable for application specific filters where the coefficients are fixed and hence not suitable for reconfigurable filters[21].

For the implementation of constant multiplications using addition/ subtraction and shift operations, a straight forward method, generally known as the digit-based recoding [7], initially defines the constants in multiplications in binary representation. 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 using the digit-based recoding method [7]. The decompositions of 29x and 43x in binary are listed as follows:

29x = (11101)binx = x>>4+x>>3+x>>2+x

43x = (101011)binx = x>>5+x>>3+x>>1+x

and require 6 addition.

However, the implementation of constant multiplications in a shift-adds architecture enables the sharing of common partial products among the constant multiplications, that significantly reduces the area and power dissipation of the MCM design. Hence, the MCM problem[1] is defined as finding the minimum number of addition/ subtraction operations that implement the constant multiplications, since shifts can be realized using only wires in hardware. Note that the MCM problem is an NP-complete problem [4]. The algorithms designed for the MCM problem can be categorized in two classes as Common Sub expression Elimination (CSE)[1] methods and graph-based (GB) techniques. While the maximization of the partial product sharing is common in these algorithms, they differ in the search space that they explore. The CSE algorithms [1, 13] initially define the constants under a particular number representation namely, binary, Canonical Signed Digit (CSD)[11],or Minimal Signed Digit (MSD), and then, find the “best" sub expression, generally the most common, among the constant multiplications. The GB algorithms [2, 6, 14] are not restricted to any particular number representation and consider a large number of alternative implementations of a constant multiplication, yielding better solutions than the CSE algorithms as shown in [2, 14]. Returning to our example in Figure 2, the exact CSE algorithm [1] gives a solution with 4 operations by finding the most common partial products 3x = (11)binx and 5x = (101)binx when constants are defined under binary, (Figure 2(b)). The exact GB algorithm [2].

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[1], 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.

II. REVIEW OF PREVIOUS METHODS

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 [18] is a signed representation.

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.

The algorithms designed for the MCM problem can be

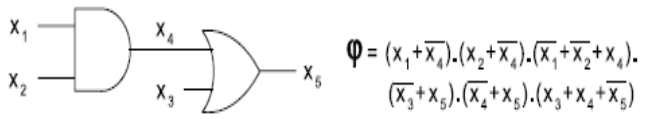

Figure 3: Example for combinational circuit with its CNF formula

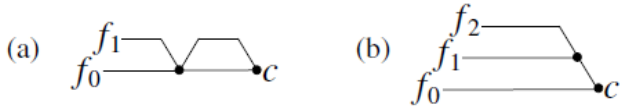

Figure 4: The coefficient *c* is obtained from (a) two existing fundamentals or (b) three existing fundamentals.

categorized in two classes: common subexpression elimination(CSE) algorithms [9] and graph-based (GB) techniques [12]. The CSE algorithms initially extract all possible subexpressions from the representations of the constants when they are defined under binary, canonical signed digit (CSD) , or minimal signed digit (MSD) [8]. Then, they find the "best" sub expression, generally the most common, to be shared among the constant multiplications. The GB methods are not limited to any particular number representation and consider a larger number of alternative implementations of a
constant, yielding better solutions than the CSE algorithms, as shown in [11] and [12].

The exact CSE algorithm of [9] gives a solution with four operations by finding the most common partial products

$3x = (11)_{\text{bin}}x$ and $5x = (101)_{\text{bin}}x$

when constants are defined under binary, as illustrated in Fig. 2(b). On the other hand, the exact GB algorithm [12] finds a solution with the minimum number of operations by sharing the common partial product $7x$ in both multiplications, Note that the partial product

$7x = (111)_{\text{bin}}x$

cannot be extracted from the binary representation of $43x$ in the exact CSE algorithm [9].
However, all these algorithms assume that the input data x is processed in parallel. On the other hand, in digit-serial arithmetic, the data words are divided into digit sets, consisting of d bits, that are processed one at a time [13]. Since digit serial operators occupy less area and are independent of the data word length, digit-serial architectures offer alternative low complexity designs when compared to bit-parallel architectures.

However, the shifts require the use of D flip-flops, as opposed to the bit-parallel MCM design where they are free in terms of hardware. Hence, the high-level algorithms should take into account the sharing of shift operations as well as the sharing of addition/subtraction operations in digit-serial MCM design. Furthermore, finding the minimum number of operations realizing an MCM operation does not always yield an MCM design with optimal area at the gate level [15]. Hence, the high-level algorithms should consider the implementation cost of each digit-serial operation at the gate level.

B. Boolean Satisfiability

Conjunctional 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. if a literal of a clause assumes value 1, then the clause is satisfied. 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 derivation of CNF formulas of basic logic gates can be found in [19].

A *combinational circuit* is a directed acyclic graph with nodes corresponding to logic gates and directed edges corresponding to wires connecting the gates. Incoming edges of a node are called *fanins* and outgoing edges are called *fanouts*.

Consider the combinational circuit and its CNF formula given in Fig. 3. 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. Here 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.

C. Digit-Serial Multiplication(20).

A new algorithm is a hybrid of the RAG-*n* (Reduced Adder Graph) and RSAG-*n*(Reduced Shift and Add Graph) algorithms. RASG-*n* work with odd coefficients like RAG-*n* and only realizes one coefficient in each iteration, like RSAG-*n*. These algorithms are graph based. Node values are referred to as fundamentals. Realized coefficients are removed from the coefficient set and added to an interconnection table that specifies how the value is obtained. The termination condition of the algorithm is that the coefficient set is empty. The steps in the
RSAG-*n* algorithm are as follows:

1.Divide even coefficients by two until odd, and save the number of times each coefficient is divided. These shifts at the outputs can be considered to be free when other coefficients are synthesised.
2. Remove zeros, ones, i.e., coefficients which corresponds to a power-of-two, and repeated coefficients from the coefficient set.
3. Compute the single-coefficient adder cost for each coefficient, which is done by using a look-uptable.
4. Compute a sum matrix based on power-of-two multiples of the fundamental values included in the interconnection table. At start this matrix is and is then extended when new fundamentals are added. If any required

$$\begin{array}{c|cccccc} & 1 & -1 & 2 & -2 & 4 & \dots \\ \hline 1 & 2 & 0 & 3 & -1 & 5 & \dots \\ -1 & 0 & -2 & 1 & -3 & 3 & \dots \\ 2 & 3 & 1 & 4 & 0 & 6 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{array}$$

coefficients are found in the matrix, compute the required number of shifts. Find the coefficients which require the lowest number of additional shifts, and select the smallest of those. Add this coefficient to the interconnection
Table and remove it from the coefficient set.
5. Repeat step 4 until no required coefficient is found in the sum matrix.
6. for each remaining coefficient, check if it can be obtained by the strategies illustrated in Fig. 4. For both cases two new adders are required. If any coefficients are found, select the smallest coefficient of those which require the lowest number of additional shifts. Add this coefficient and the extra fundamental to the interconnection table. Remove the coefficient from the coefficient set.
7. Repeat step 5 and 6 until no required coefficient is found.
8. Choose the smallest coefficient with lowest single coefficient adder cost. Different sets of fundamentals that can be used to realize the coefficient are obtained from a look-up-table. For each set, remove fundamentals that are already included in the interconnection table and compute the required
number of shifts. Find the sets which require the lowest number of additional shifts, and of those, select the set with smallest sum. Add this set and the coefficient to the interconnection table.
Remove the coefficient from the coefficient set.
9. Repeat step 5, 6, 7, and 8 until the coefficient set is empty. The basic ideas for the RAG-*n* [5], RSAG-*n* [10], and RASG-*n* algorithms are similar, but the resulting difference is significant. The main difference between
the first two algorithms is that RAG-*n* chooses to realize coefficients by using extra fundamentals of minimum value, while RSAG-*n* chooses fundamentals that require a minimum number of shifts. The result of these two different strategies is that RAG-*n* is more likely to reuse fundamentals, due to the selection of smaller fundamental values and by that reduce the adder cost, while RSAG-*n* is more likely to reduce the number of shifts. As the proposed algorithm, RASG-*n*, is a hybrid of these strategies realizations with both few adders and few shifts are obtained. It is worth noting that if all coefficients are

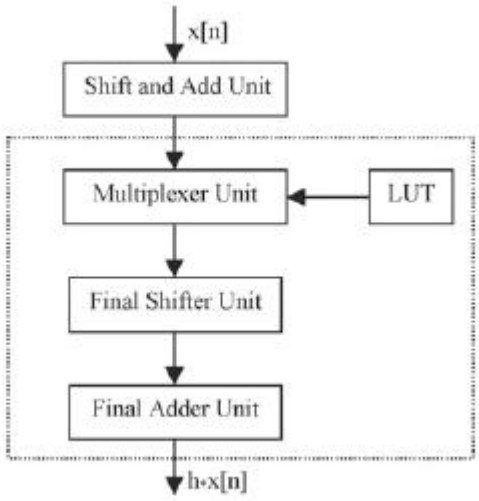

Figure 5: Architecture of proposed method.

realized before step 6 of the algorithm, the implementation has optimal adder cost

II. Review of BCSE Method

This[6] deals with the elimination of redundant binary common subexpressions (BCSs) that occur within the coefficients. The BCSE technique focuses on eliminating redundant computations in coefficient multipliers by reusing the most common binary bit patterns (BCSs) present in coefficients.
An *n*-bit binary number can form $2n - (n + 1)$ BCSs among themselves. For example, a 3-bit binary representation can form four BCSs, which are [0 1 1], [1 0 1], [1 1 0], and [1 1 1]. These BCSs can be expressed as [0 1 1] = $x2 = 2{-}1x + 2{-}2x$, [1 0 1] = $x3 = x + 2{-}2x$, [1 1 0]= $x4 = x + 2{-}1x$, and [1 1 1]= $x5 = x + 2{-}1x + 2{-}2x$, where x is the input signal. Note that other BCSs such as [0 0 1], [0 1 0], and [1 0 0] do not require any adder for implementation as they have only one nonzero bit. A straightforward realization of above BCSs would require five adders. However $x2$ can be obtained from $x4$ by a right shift operation (without using any extra adders): $x2 = 2{-}1x + 2{-}2x = 2{-}1(x + 2{-}1x) = 2{-}1x4$. Also, $x5$ can be obtained from $x4$ using an adder: $x5 = x + 2{-}1x + 2{-}2x = x4 + 2{-}2x$.
Thus, only three adders are needed to realize the BCSs $x2$ to $x5$. The number of adders required for all the possible *n*-bit binary sub expressions is $2n{-}1 - 1$ [6]. The number of adders needed to implement the
Coefficient multipliers using the binary representation-based BCSE is considerably less than the CSD-based CSE methods [6]. The proposed FIR filter architecture is based on transposed direct form as shown in Fig. 1. In the transposed direct form, the coefficient multipliers (shown as dotted outline in Fig. 1) share the same input and hence commonly known as multiplier block (MB). The MB reduces the

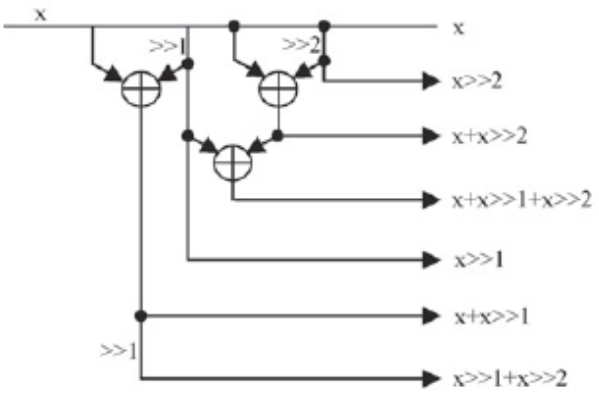

Figure 6: Architecture of shift and add unit.

complexity of the FIR filter implementations, by exploiting the redundancy in MCM. Thus, redundant computations (partial product additions in the multiplier) are eliminated using BCSE. The BCSE method in [6] was formulated as a low complexity solution to realize application specific filters where the coefficients are fixed.

In the case of channel filters for SDR receivers, the coefficients need to be changed as the filter specification changes with the communication standard. Therefore, reconfigurability is a necessary requirement for SDR channel filters. In the next section, we propose two architectures that incorporate reconfigurability into the BCSE-based low complexity filter architecture. We use CSD[11] to illustrate proposed reconfigurable filter architectures[21] in this paper, it must be noted that the proposed architectures can be used for any CSE method with appropriate modifications.

III. Proposed Filter Architectures

In this section, the architecture of the proposed FIR filter is presented. Our architecture is based on the transposed direct form FIR filter structure as shown in Fig. 1. The dotted portion in Fig. 1 represents the MB. In Fig. 1, PE-*i* represents the processing element corresponding to the *i*th coefficient. PE performs the coefficient multiplication operation with the help of a shift and add unit which will be explained in the latter part of this section. The architecture of PE is different for proposed CSM and PSM. In the CSM, the filter coefficients are partitioned into fixed groups and hence the PE architecture involves constant shifters. But in the PSM, the PE consists of programmable shifters (PS). The FIR filter architecture can be realized in a serial way in which the same PE is used for generation of all partial products by convolving the coefficients with the input signal ($h * x[n]$) or in a parallel way, where parallel PE architectures are employed. The first option is used when power consumption and area are of prime concern. The basic architecture of the PE (dotted portion) is shown in Fig. 5 . The functions of different blocks of the PE are explained below.

1) Shift and Add Unit*:*

It is well known that one of the efficient ways to reduce the complexity of multiplication operation is to realize it using shift and add operations. In contrast to conventional shift and add units used in previously proposed reconfigurable filter architectures, we use the BCSs-based shift and add unit in our proposed CSM and PSM architectures. The architecture of shift and add unit is shown in Fig. 3. The shift and add unit is used to realize all the 3-bit BCSs of the input signal ranging from [0 0 0] to [1 1 1]. In Fig. 3, "$x>>k$" represents the input x shifted right by k units. All the 3-bit BCSs [0 1 1], [1 0 1], [1 1 0], and [1 1 1] of a 3-bit number are generated using only three adders, whereas a conventional shift and add unit would require five adders. Since the shifts to obtain the BCSs are known beforehand, PS are not required. All these eight BCSs (including [000]) are then fed to the multiplexer unit. In both the architectures (CSM and PSM) proposed in this paper, we use the same shift and add unit. Thus, the use of 3-bit BCSs reduces the number of adders needed to implement the shift and add unit compared to conventional shift and add units.

2) Multiplexer Unit*:*

The multiplexer units are used to select the appropriate output from the shift and add unit. All the multiplexers will share the outputs of the shift and add unit. The inputs to the multiplexers are the 8/4 inputs from the shift and add unit and hence 8:1/4:1 multiplexer units are employed in the architecture. The select signals of the multiplexers are the filter coefficients which are previously stored in a look up table (LUT). The CSM and PSM architectures basically differ in the way filter coefficients are stored in the LUT. In the CSM, the coefficients are directly stored in LUTs without any modification whereas in PSM, the coefficients are stored in a coded format. The number of multiplexers will also be different for PSM and CSM. In CSM, the number of multiplexers will be dependent on the number of groups after the partitioning of the filter coefficient into fixed groups. The number of multiplexers in the PSM is dependent on the number of non-zero operands in the coefficient for the worst case after the application of BCSE algorithm.

3) Final Shifter Unit*:*

The final shifter unit will perform the shifting operation after all the intermediate additions (i.e., intra-coefficient additions) are done. This can be illustrated using the output expression

$$y = 2^{-4}x + 2^{-6}x + 2^{-15}x + 2^{-16}x. \quad (1)$$

By coefficient-partitioning [16], we obtain

$$y = 2^{-4}(x + 2^{-2}x) + 2^{-15}(x + 2^{-1}x). \quad (2)$$

After obtaining the intermediate sums (x + 2−2x) and (x + 2−1x) from the shift and add units with the help of multiplexer unit, the final shifter unit will perform the shift operations 2−4and 2−15 in (2). The PSM and CSM architectures also differ in the nature of final shifters. In the CSM, the final shifts are constants and hence no PS are required. In the PSM, we have used PS.

4) Final Adder Unit*:*

This unit will compute the sum of all the intermediate additions 2−4(x + 2−2x) and 2−15(x + 2−1x) as in (2). As the filter specifications of different communication standards are different, the coefficients change with the standards. In conventional reconfigurable filters, the new coefficient set corresponding to the filter specification of the new communication standard is loaded in the LUT. Subsequently, the shift and add unit performs a bitwise addition after appropriate shifts. On the contrary, the proposed CSM and PSM architectures perform a binary common subexpression (BCS)- wise addition (instead bitwise addition). Thus, the same hardware architecture can be used for different filter specifications to achieve the necessary reconfigurability. Moreover, the proposed BCS-based shift and add unit reduces addition operations and hence offers hardware complexity reduction. In the next section, the CSM is explained in a detailed manner.

IV. Architecture of PSM

The PSM is based on the BCSE algorithm presented in our previous work [6]. The PSM architecture presented in this section incorporates reconfigurability into BCSE. The PSM has a pre-analysis part in which the filter coefficients are analyzed using the BCSE algorithm in [6]. Thus, the redundant computations (additions) are eliminated using the BCSs and the resulting coefficients in a coded format are stored in the LUT. The coding format is explained in the latter part of this section. The shift and add unit is identical for both PSM and CSM. The number of multiplexer units required can be obtained from the filter coefficients after the application of BCSE [6]. The number of multiplexers is selected after considering the number of non-zero operands (BCSs and unpaired bits) in each of the coefficients after the application of the BCSE algorithm. The number of multiplexers will be corresponding to the number of non-zero operands for the worst-case coefficient (worst-case coefficient being defined as

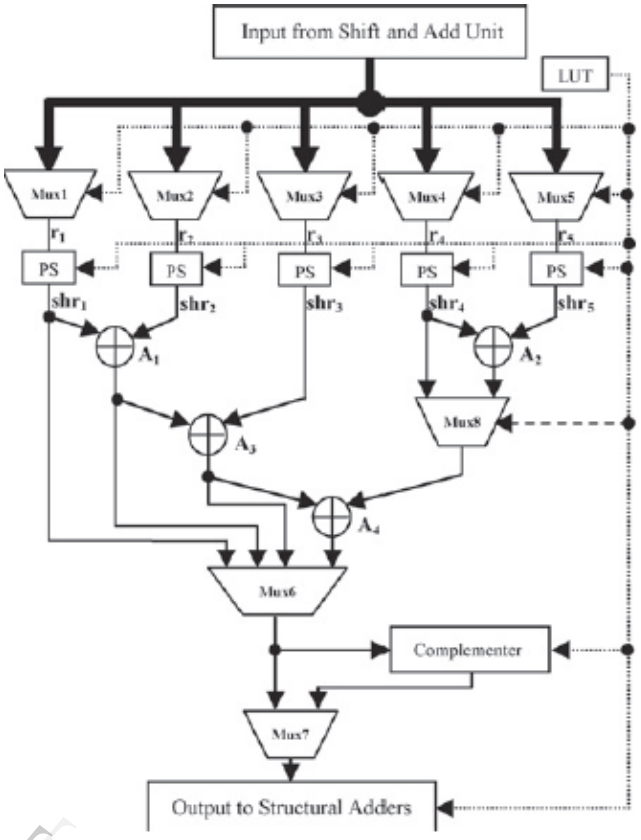

Figure 7: Architecture of PE for PSM.

coefficient that has the maximum number of non-zero operands).

The architecture of PE for PSM is shown in Fig. 5. The coefficient wordlength is fixed as 16 bits. We have done the statistical analysis for various filters with coefficient precision of 16 bits and different filter lengths (20, 50, 80, 120, 200, 400, and 800 taps) and it was found that the maximum number of non-zero operands is 5 for any coefficient. The analysis was done for filters with different passband (ωp) and stopband (ωs) frequency specifications given by (1) is $\omega p = 0.1\pi$, $\omega s = 0.12\pi$; 2) $\omega p = 0.15\pi$, $\omega s = 0.25\pi$; 3) $\omega p = 0.2\pi$, $\omega s = 0.22\pi$; and 4) $\omega p = 0.2\pi$, $\omega s = 0.3\pi$, respectively.

Based on our statistical analysis, we have fixed the number of multiplexers as 5 (same as the number of non-zero operands). The LUT consists of two rows of 18 bits for each coefficient of the form SDDDDXXDDDDXXMMMML and DDDDXXDDDDXXDDDDXX, where "S" represents the sign bit, "DDDD" represents the shift values from 20 to 2−15 and "XX" represents the input "x" or the BCSs obtained from the shift and add unit. In the coded format, *XX* = "01" represents "x," "10" represents x+2−1x, "11" represents x + 2−2x, and "00" represents x + 2−1x + 2−2x, respectively.

Thus, the two rows can store up to five operands which is the worst case number of operands for a 16-bit coefficient. In most of the practical coefficients, the number of operands is less than the worst case number of operands, 5. In that case

"MMMML" can be used to avoid unnecessary additions. The values "MMMM" will be given as select signal to the Mux6 and "L" to Mux8. "MMMML" indicates the presence of five operands. A "1" in each position indicates the presence of each operand. Thus, for all operands to be present will be indicated by "MMMML" = "11111." This means the Mux6 will select the output from the output of adder, A4 and Mux8 will select the output of adder, A2. If only first operand is present, "MMMML" = "10 000." This means the Mux8 will select the output of PS, shr4 and Mux6 will select the output of PS, shr1. As a result of this none of the adders shr1 to shr4 will be loaded saving significant amount of dynamic power. The coding can be explained as given below. Consider the positive coefficient *h* operands. A "1" in each position indicates the presence of each operand. Thus, for all operands to be present will be indicated by "MMMML" = "11111." This means the Mux6 will select the output from the output of adder, A4 and Mux8 will select the output of adder, A2. If only first operand is present, "MMMML" = "10 000." This means the Mux8 will select the output of PS, shr4 and Mux6 will select the output of PS, shr1. As a result of this none of the adders shr1 to shr4 will be loaded saving significant amount of dynamic power.

The coding can be explained as given below. Consider the positive coefficient *h*

$$h = [1010011001010011]. \quad (3)$$

By using the BCSE [6], substituting 2 = [1 1], 3 = [1 0 1], (11) becomes

$$h = [3000020003000020]. \quad (4)$$

Then (12) will be stored in the LUT as 0000011010011011110 and 100111111010000000. It must be noted that as (12) has only four operands, the fifth operand values "DDDDXX" are substituted as 000000 and "MMMML" as "11110." The XX values are given as select signals for Mux1 to Mux5. The values of DDDD are fed to corresponding PS. The multiplexer Mux6 and Mux8 will select the appropriate output in case the number of operands after BCSE is less than 5. The use of Mux6 and Mux8 reduces the number of adders utilized by selecting the output from the appropriate adder as all the adders in the PE are not always needed. For example, in (12), as only four operands occur, output can be taken from the output of PS, shr4 without using adder, shr2. Mux8 will do this and hence the adder shr2 is not loaded and consumes zero current and power. The select signals of Mux6 and Mux8 have five bits and hence 25 different control signals are possible which adds lots of flexibility to the architecture which can be employed in future if required. Mux7 is used to complement the output in case of a negative coefficient and its select signal is the sign bit "S" of the coefficient.

The PSM architecture has two advantages;

TABLE1: Synopsys Synthesis Results for 20-Tap FIR Filter Implementation

	Binary programmable shifts method (BPSM)	Binary constant shifts method (BCSM)	CSD-CSM	CSD-PSM	FIR Filter [15]
Area (mm^2)	0.2594	0.275	0.304	0.2796	0.5467
Delay (ns)	8.2	7.67	8.5	9.34	15.6
Dynamic power (mW)	5.98	7.8	10	13.97	16

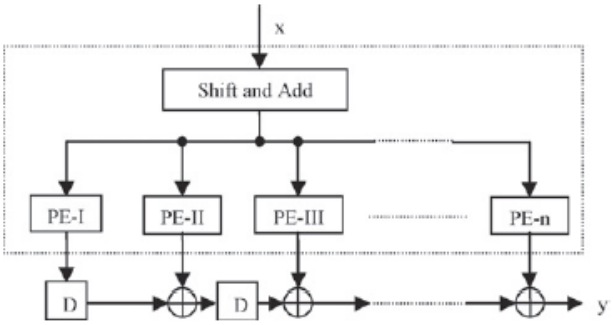

Figure 8:Transposed direct form of an FIR filter.

first, it guarantees a reduced number of additions compared to CSM, and second it offers the flexibility of changing the wordlength of coefficients. The same PSM architecture designed for 16-bit coefficients is capable of operating for any coefficient wordlength less than 16 bits. This means, if the wordlength is reduced, the format of the LUT can be changed if required. The main advantage of reducing the precision is that some of the adders in the PSM architecture will be unloaded resulting in zero dynamic power. To the best of our knowledge, the PSM architecture is the first approach toward programmable coefficient wordlength FIR filter architecture. This means that the coefficient wordlength of the proposed PSM architecture can be changed dynamically without any change in hardware. Hence transposed direct form posses the architecture as shown in figure 8.

V. Experimental Results

In this section, the synthesis and design results of the proposed CSM and PSM architectures are presented and compared with the recently proposed reconfigurable FIR filter architectures in the literature [11], [12], [14], [15], [20], [21

A. Synthesis Results

We have used Xilinx 8.1i ISE for synthesizing purposes. The synthesis has been done on Xilinx's Spartan-3E FPGA. Table I shows the synthesis results of the CSM and

PSM 20-tap FIR filter that has a coefficient wordlength of 16 bits. We have done the implementation of filters with different passband edge (ωp) and stopband edge (ωs) specifications.

Even though the proposed architectures are known to reconfigurable, the usage of adders and shifters is dependent on the filter coefficient values. Some of the adders may not be used by the multiplexers. As a result of this, they are unloaded and do not consume any dynamic power. Hence, the power and speed values of the synthesis results are dependent on the filter coefficients and hence we have considered an average of the synthesis results in all the tables in this paper. From the comparison it is very evident that the CSM requires 475 gates more than that of PSM, whereas PSM requires 6.82 ns more for the data to arrive at the output compared to CSM. Thus, the CSM results in higher speed whereas the PSM results in lower area. The reason for lower speed of PSM is due to the presence of programmable shifters and that of less area
is due to elimination of redundant additions by using BCSE algorithm. We have also analyzed the effect of the MB for different filter coefficient word lengths of 8, 12, and 16 bits for the PSM architecture.

B.CSD Based Reconfigurable FIR Filter Architecture

CSD based CSE algorithms are considered to be one of the best algorithms that can result in low complexity fixedcoefficient FIR filter implementations. However to the best of our knowledge, the implementation of the CSD-CSE based reconfigurable filter architectures has not been addressed in the literature. We have implemented a CSD based FIR filter using the CSM architecture (CSD-CSM) and a CSD-CSE based FIR filter using the PSM architecture (CSD-PSM). For low complexity, we have employed the CSE algorithm in [3] on the coefficients before they are stored in LUT. We have implemented
a CSD based shift and add unit to generate common subexpression (CSs) such as [1 0 1], [1 0 −1], [1 0 0 1] and [1 0 0 −1] and their negated versions. In the previous works based on CSE algorithm [3]–[5], it was considered
that common subexpressions (CSs) such as [−1 0 − 1] and [−1 0 1] can be generated from their respective negated
versions [1 0 1] and [1 0 − 1] without using any extra adder
by configuring the existing adder as a subtractor. But this is applicable only for fixed coefficient filters. An *n*-bit adder circuit would require *n* additional XOR gates to reconfigure the adder to subtractor mode. These additional XOR gates would increase the critical path of the adder circuit (equivalent to the delay imposed by *n*
half-adders) and impose overheads for CSD implementation of the FIR filter. Another drawback
of CSD implementation is with the storage of coefficients in LUT. The CSD value like [1 0 − 1 0 − 1 0 1 0 − 1] can be
stored in an LUT like [01 00 11 00 11 00 01 00 11] with "00" corresponding to 0, "01" corresponding to 1, and "11" corresponding to −1. Therefore, for the worst-case scenario, an 8-bit CSD coefficient requires 16 bits for its representation.

This can be optimized as no adjacent bits in CSD are ones. But still CSD requires more number of bits than binary. Since all the bits in binary representation are positive, this problem will not come. Thus, the additional half-adders required for implementing the dder/subtractor circuit and the additional storage space required for CSD will increase the area and the additional half-adders in the adder/subtractor unit reduces the speed of operation of the CSD based reconfigurable FIR filters compared to binary based FIR filter implementations.

This becomes highly significant, as the order of the channel filters in wireless communication transceivers is very high. We have done the synthesis using Synopsys tool. The synthesis results for a 20-tap FIR filter with 16-bit coefficient wordlength are summarized in Table 1. The proposed CSM and PSM architectures which employ binary representation of filter coefficients are denoted as BCSM and BPSM, respectively. The CSD based implementations of CSM and PSM are denoted as CSD-CSM and CSD-PSM, respectively. Table I shows that the CSDCSM and CSD-PSM architectures consume more area, power, and has less speed compared to our binary representation based BPSM and BCSM architectures. The BCSM architecture has area reduction of 10% and 1% over CSD-CSM and CSD-PSM architectures, respectively, and the area reduction for BPSM architecture over CSD-CSM and CSD-PSM architectures are 15% and 7%, respectively. The improvement in the speed of operation for the BCSM architecture over the CSD-CSM and CSD-PSM architectures is 10% and 22%, respectively. The
BPSM architecture offers an improvement in the speed of operation of 4% and 12% over the CSD-CSM and CSD-PSM architectures, respectively. The dynamic power reductions for the BCSM architecture are 22% and 44% over the CSD-CSM and CSD-PSM architectures, respectively. The BPSM architecture[21] offers the dynamic power reductions of 40% and 57% over the CSD-CSM and CSD-PSM[21] architectures, respectively. The BPSM architecture offers area and power reductions of 6% and 23% over the BCSM architecture, respectively. The BCSM architecture offers in improvement in the speed of operation by 7% compared to the BPSM architecture. In Table 1, the proposed architectures are also compared with the MMCM architecture based FIR filter in [15]. The BCSM architecture offers an area reduction of 49.7%, power reduction of 51.3%, and a speed improvement of 50.8% over the MMCM [15]. The area and power reductions offered by the BPSM architecture over MCM [15] are 52.7% and 62.5%, respectively.

VI. Implementation Results

We have implemented the proposed CSM and PSM architectures for a 20-tap FIR filter with 16-bit coefficient precision on Xilinx's Spartan-3E FPGA associated with
the dual DSP-FPGA Signal master kit . The PSM also provides the flexibility of changing the filter coefficient

wordlengths dynamically. Using CSD, the worst case cost remains b2 + O(1), but the average case is now improved to b3 + O(1. The optimal decomposition in terms of add/subtract operations is in general not obtained with CSD, and its worst case and average costs are unknown.

VII. CONCLUSION

In this paper, designing of digit-serial MCM operation with optimal area is done. Also a new reconfigurable architecture using PSM is proposed which provides the flexibility of changing the filter coefficient word lengths dynamically. The experimental results indicate that the complexity of digit-serial MCM designs can be further reduced using the high-level optimization algorithms proposed in this paper.

VIII. REFERENCES

[1] Levent Aksoy, Cristiano Lazzari, Eduardo Costa, Paulo Flores, and José Monteiro, **"**Design of Digit-Serial FIR Filters: Algorithms, Architectures, and a CAD Tool" ,*IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 21, NO. 3, MARCH 2013*

[2] J. Mitola, "Object-oriented approaches to wireless systems engineering," in *Software Radio Architecture*. New York: Wiley, 2000.

[4] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova,
"A new algorithm for elimination of common subexpressions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 1, pp. 58–68, Jan. 1999.

[6] R. Mahesh and A. P. Vinod, "A new common subexpression elimination
algorithm for realizing low complexity higher order digital filters," *IEEE
Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp.
217–219, Feb. 2008

[7] A. P. Vinod and E. Lai, "Low power and high-speed implementation of FIR filters for software defined radio receivers,"*IEEETrans.Wireless*

[8] I.-C. Park and H.-J. Kang, "Digital filter synthesis based on minimal
signed digit representation," in *Proc. 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] X. Chenghuan, C. He, Z. Shunan, and W. Hua, "Design and implementation of a high-speed programmable polyphase FIR filter," in *Proc.5th Int. Conf. Applicat.-Specific Integr. Circuit*, vol. 2. Oct. 2003, pp. 783–787.

[14] S. S. Demirsoy, I. Kale, and A. G. Dempster, "Efficient
implementation of digital filters using novel reconfigurable multiplier blocks," in *Proc. 38th Asilomar Conf. Signals Syst. Comput.*, vol. 1. Nov. 2004, pp. 461–464.

[15] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Optimization of area
in digital FIR filters using gate-level metrics," in *Proc. DAC*, 2007, pp.
420–423.

[18] A. Avizienis, "Signed-digit number representations for fast parallel
arithmetic," *IRE Trans. Electron. Comput.*, vol. 10, no. 3, pp. 389–400,
Sep. 1961.

[19] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE
Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 1, pp.
4–15,Jan.1992.

[20] K. Johansson, O. Gustafsson, and L. Wanhammar, "Multiple constant
multiplication for digit-serial implementation of low power FIR filters,"
WSEAS Trans. Circuits Syst., vol. 5, no. 7, pp. 1001–1008, 2006.

[21] R. Mahesh, A. P. Vinod "New Reconfigurable Architectures for Implementing FIR Filters with Low Complexity", *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 29, NO. 2, FEBRUARY 2010.*