International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 7, Issue 7 (June 2013), PP. 69-76

Hardware Efficient Reconfigurable FIR Filter

Balu I[1], Venkatesan.K[2]
[1]Assistant Professor(PG Scholar), Department of Electronics and Communication, Sri Shakthi Institute of Engineering and Technology,Coimbatore.
[2]Assistant Professor, Department of Electronics and Communication, Sri Shakthi Institute of Engineering and Technology Coimbatore.

Abstract:- Reconfigurability and low complexity are the two key requirements of finite impulse response (FIR) filters employed in multi-standard wireless communication systems. In this paper, two new reconfigurable architectures of low complexity FIR filters are proposed, namely constant shifts method and programmable shifts method. The proposed FIR filter architecture is capable of operating for different word length filter coefficients without any overhead in the hardware circuitry. We show that dynamically reconfigurable filters can be efficiently implemented by using common sub-expression elimination algorithms. Process architecture has been implemented in Xilinx Spartan-3E FPGA

Keywords:- Common Sub-expression Elimination,FIR filter, high level synthesis, Re-configurability, FPGA

I. INTRODUCTION

Digital FIR filters find extensive applications in mobile communication systems for applications such as channelization, channel equalization, matched filtering, and pulse shaping, due to their absolute stability and linear phase properties. The filters employed in mobile systems must be realized to consume less power and operate at high speed. Recently, with the advent of software defined radio (SDR) technology, finite impulse response (FIR) filter research has been focused on reconfigurable realizations. The fundamental idea of an SDR is to replace most of the analog signal processing in the transceivers with digital signal processing in order to provide the advantage of flexibility through reconfiguration. This will enable different air-interfaces to be implemented on a single generic hardware platform to support multi-standard wireless communications. Wideband receivers in SDR must be realized to meet the stringent specifications of low power consumption and high speed. Reconfigurability of the receiver to work with different wireless communication standards is another key requirement in an SDR. The most computationally intensive part of an SDR receiver is the channelizer since it operates at the highest sampling rate. It extracts multiple narrowband channels from a wideband signal using a bank of FIR filters, called channel filters. Using polyphase filter structure, decimation can be done prior to channel filtering so that the channel filters need to operate only at relatively low sampling rates. This can relax the speed of operation of the filters to a good extent. However due to the stringent adjacent channel attenuation specifications of wireless communication standards, higher order filters are required for channelization and consequently the complexity and power consumption of the receiver will be high. As the ultimate aim of the future multi-standard wireless communication receiver is to realize its functionalities in mobile handsets, where its full utilization is possible, low power and low area implementation of FIR channel filters is inevitable. In, the filter multiplications are done via state machines in an iterative shift and add component and as a result of this there is huge savings in area. For lower order filters, the approach in offers good trade-off between speed and area. But in general, the channel filters in wireless communication receivers need to be of high order to achieve sharp transition band and low adjacent channel attenuation requirements. For such applications, the approach in results in low speed of operation. 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) coefficients produce low complexity FIR filter coefficient multipliers. The goal of CSE is to identify multiple occurrences of identical bit patterns that are present in the CSD representation of coefficients, and eliminate these redundant multiplications. A modification of the 2-bit CSE technique in [3] for identifying the proper patterns for elimination of redundant computations and to maximize the optimization impact was proposed in [4]. The technique in [3] was modified 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. We have proposed the binary common sub-expression elimination (BCSE) method which provided improved adder reductions.

In, a high speed programmable CSD based FIR filter was proposed. The filter architecture consisted of a programmable CSD based Booth encoding scheme and partial product Wallace adder tree. The final adder was a carry look-ahead adder. Though this method offered a high speed solution, the resulting filters consume more

power. However, this method used the built-in block multipliers of Virtex II field-programmable gate array (FPGA) and there was no consideration for the complexity reduction of the FIR filter. The concept of reconfigurable multiplier block (ReMB) was introduced. The ReMB will generate all the coefficient products and a multiplexer will select the required ones depending on the input. It was shown that by pushing the multiplexer deep into the multiplier block design, the redundancy can be reduced. The resulting specialized multiplier design can be more efficient in terms of area and computational complexity compared to the general-purpose multiplier plus the coefficient store. But the ReMB proposed in has its area, power, and speed dependent on the filter-length making them inappropriate for higher order FIR filters. In multiplexed multiple constant multiplications (MMCM) approach was proposed. This method considers the coefficient set as a constant and uses the graph dependence (GD) algorithms for reducing redundancy. But this method follows a directed acyclic graph structure which will result in long LD and thus lower speed of operation as reported in [3], [4]. Also the area of the architecture linearly increases with the filter length as in filters with filter-length above 40 are infeasible. In the common digital signal processing (DSP) operations such as filtering and matrix multiplication were identified and expressed as vector scaling operations. In order to apply vector scaling, simple number decomposition strategies were identified. The idea was to precompute the values such as x, $3x$, $5x$, $7x$, $9x$, $11x$, $13x$, and $15x$, where x is the input signal and then reuse these precomputations efficiently using multiplexers. The presence of multiplexers gave the option of adaptive computing for the method in. In the method was modified and efficient circuit-level techniques, namely a new carry select adder and conditional capture flip-flop, were used to further improve power and performance. The architectures are appropriate only for relatively lower order filters and hence not suitable for channel filters in communication receivers. Although a few works addressed the problem of reducing the complexity of coefficient multipliers in reconfigurable FIR filters, hardly any work demonstrated reconfigurability in higher order filters. Moreover, we note that there is sufficient scope for more work on complexity reduction in reconfigurable filters especially for wireless communication applications where higher order filters are often required to meet the stringent adjacent channel attenuation specifications.

In this paper, we propose two architectures that integrate reconfigurability and low complexity to realize FIR filters.

The FIR filter architectures proposed are called constant shifts method (CSM) and programmable shifts method (PSM). We have presented the preliminary design of these architectures in a recent conference paper. In this journal, we elaborate the CSM and PSM architectures introduced in by providing the detailed design. The design analysis of the architectures and their extension to high-level synthesis are presented. The proposed architectures have been synthesized on 0.18μm complementary metal-oxide-semiconductor (CMOS) technology and compared with the recent methods. Also we have implemented two CSD based methods based on our CSM and PSM to compare the complexities of the CSD and binary based CSE techniques. The proposed architectures consider coefficients as constants (as they are stored in LUTs) and input signal as variable.

The coefficient multiplication in such a case is known as multiple constant multiplications (MCM), i.e., multiplication of one variable (input signal) with multiple constants (filter coefficients). The MCM is then optimized for eliminating redundancy using our recently proposed BCSE algorithm [5] to minimize the filter complexity. The proposed CSM focuses on the implementing FIR filters by partitioning the filter coefficients into fixed groups. The PSM has a pre-analysis part which eliminates the redundancy in filter coefficients using the BCSE algorithm. The advantage of CSM is that it produces high-speed filters at the cost of a slight increase in area and power consumption. On the contrary, the PSM produces filters with low area and power consumption at the cost of a slight increase in delay. Another advantage of PSM is that the word length of the filter coefficients can be dynamically changed without any modification in the hardware.

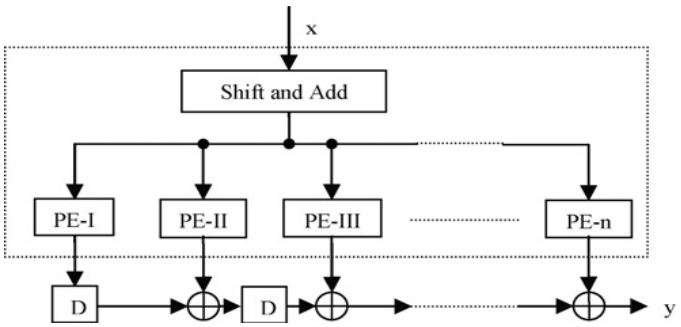

Fig. 1. Transposed direct form of an FIR filter.

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 ith 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. 2.The functions of different blocks of the PE are explained below.

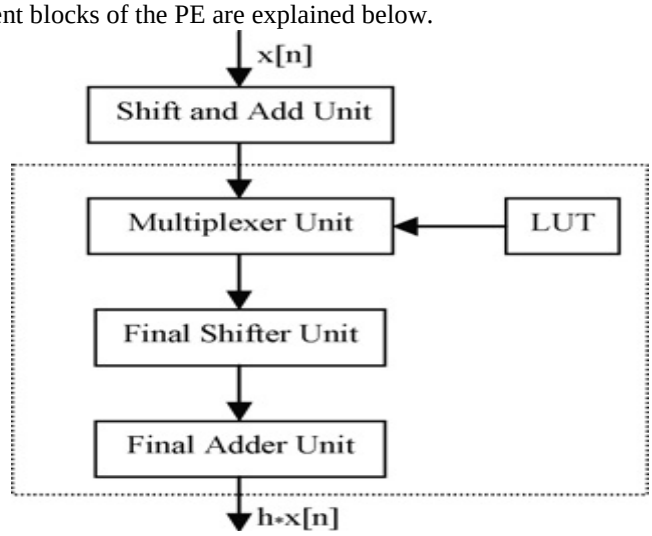

Fig 2. Architecture of proposed method.

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

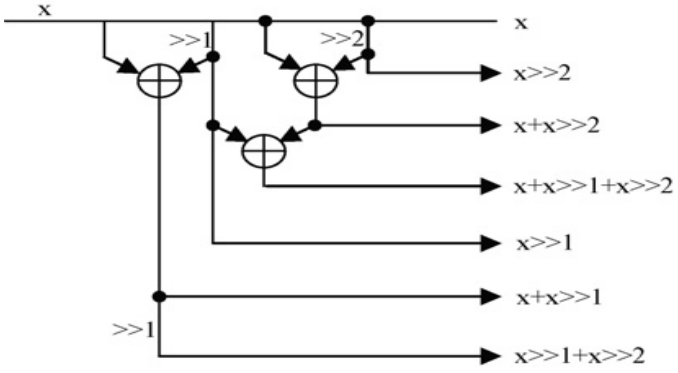

Fig 3. Architecture of shift and add unit.

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

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

By coefficient-partitioning [5], we obtain

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

II. ARCHITECTURE OF CSM

In the CSM architecture, the coefficients are stored directly in the LUT. These coefficients are partitioned into groups of 3-bits and are used as the select signal for the multiplexers. The number of multiplexer units required is n⁄3, where n is the word length of the filter coefficients. The CSM can be explained with the help of an 8-bit coefficient h = "0.11111111." This coefficient h is the worst-case 8-bit coefficient since all the bits are nonzero and hence needs a maximum number of additions and shifts. In this case, n = 8, and therefore the number of multiplexers required is 3. The output $y = h\,(x)$ is expressed as

$y = 2{-}1x+2{-}2x+2{-}3x+2{-}4x+2{-}5x+2{-}6x+2{-}7x+2{-}8x.$ (3)

By partitioning into groups of three bits from most significant bit (MSB) (3), we obtain

$h = 2{-}1(x+2{-}1x+2{-}2x+2{-}3x+2{-}4x+2{-}5x+2{-}6x+2{-}7x)$

$h = 2{-}1(x+2{-}1x+2{-}2x+2{-}3(x+2{-}1x+2{-}2x)+2{-}6(x+2{-}1x))$

Note that the terms $x + 2{-}1x + 2{-}2x$ and $x + 2{-}1x$ can be obtained from the shift and add unit. Then by using the three multiplexers (mux), two 8:1 mux for the first two 3-bit groups and one 4:1 mux for the last two bits of the filter coefficients, the intermediate sums shown inside the brackets of (5) can be obtained. The final shifter unit will perform the shift operations 2−1, 2−3, and 2−6. Since these shifts are always constant irrespective of the coefficients, programmable shifters are not required and these shifts can be hardwired. The final adder unit will compute the sum of all the intermediate sums to obtain $h\,(x[n])$.

The architecture of PE for CSM is shown in Fig. 4. The coefficient word length is considered as 16 bits. The filter coefficients are stored in the LUT in sign-magnitude form with the MSB reserved for the sign bit. The first bit after the sign bit is used to represent the integer part of the coefficient and the remaining 16 bits are used to represent the fractional part of the coefficient. Thus, each 16-bit coefficient is stored as an 18-bit value in LUTs. Each row in LUT corresponds to one coefficient. Note that only half the number of coefficients needs to be stored as FIR filter coefficients are symmetric. The coefficient values corresponding to 20 to 2−14 are partitioned into groups of three bits and are used as select signals to multiplexers Mux1 to Mux5. i.e., the set (20, 2−1, 2−2) forms the select signal to Mux1 and so on. Since there are 3-bits, eight combinations are possible and hence Mux1 to Mux5 are 8:1 multiplexers. The value corresponding to 2−15 forms the select to a 2:1 multiplexer, Mux6. The output from the ith multiplexer is denoted as ri. Note that even though we are taking coefficient with values up to a precision of 16 bits, the shifting of 2−1 is done finally as shown in (4) and (5) and hence the maximum shift will be 2−15. Mux7 determines whether the output needs to be complemented based on the sign bit of the filter coefficient and hence it is a 2:1 multiplexer. In FIR filters, coefficient values are always less than one. [In our design examples, we used the Parks–McClellan algorithm to design filters (using "firpm" command in MATLAB)].

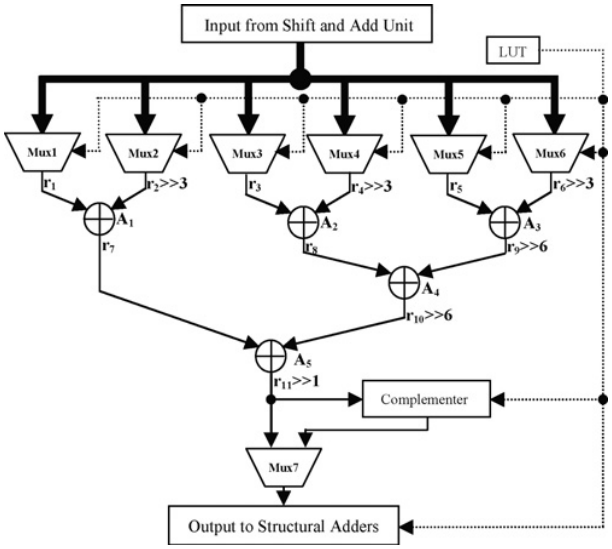

Fig. 4. Architecture of PE for CSM.

Hence, we have not employed the integer bit. However if an integer digit is required, the proposed architectures do not impose any restrictions to accommodate it.

In Fig. 4, the shifts are obtained as follows. Let $r1$ to $r6$ denotes the outputs of Mux1 to Mux6, respectively. Then

$$y = 2^{-1}r1 + 2^{-4}r2 + 2^{-7}r3 + 2^{-10}r4 + 2^{-13}r5 + 2^{-16}r6. \quad (6)$$

The shifts are obtained by partitioning the 16-bit coefficient into groups of 3-bits.

By partitioning (6)

$$y = 2^{-1}[(r1 + 2^{-3}r2) + 2^{-6}[(r3 + 2^{-3}r4) + 2^{-6}(r5 + 2^{-3}r6)]]. \quad (7)$$

Substituting $(r1 + 2^{-3}r2)$, $(r3 + 2^{-3}r4)$, and $(r5 + 2^{-3}r6)$ by $r7$, $r8$, and $r9$, respectively, we get

$$y = 2^{-1}[r7 + 2^{-6}(r8 + 2^{-6}r9)]. \quad (8)$$

By substituting $(r8 + 2^{-6}r9)$ by $r10$

$$y = 2^{-1}(r7 + 2^{-6}r10). \quad (9)$$

By substituting $(r7 + 2^{-6}r10)$ by $r11$

$$y = 2^{-1}(r11). \quad (10)$$

The expressions from (5) are represented in Fig. 4. The main advantage of the CSM architecture is that all the shifts are constants irrespective of the coefficients and hence can be hardwired resulting in high speed operation of the filter. We have employed the shift and add unit which can generate all the 3-bit BCSs using only three adders. The impact of using higher order BCSs (4-bit, 5-bit BCSs, etc.) has also been investigated. The choices of the best shifts and add unit will depend on the complexities of: 1) shift and add unit; 2) multiplexer unit; and 3) final adder unit. The number of adders needed to implement n-bit CSs is $2n-1-1$ [4]. Thus, shift and add units capable of generating 4-bit, 5-bit, and 6-bit BCSs would require 7, 15, and 31 adders, respectively. The LD is two adder-steps for both the 3-bit and the 4-bit BCS based shift and add units, and hence they have the same speed. The LD of 5-bit and 6-bit BCSs-based shift and add units are same, i.e., three adder-steps, which is one adder-step more than that of the 3-bit and 4-bit BCSs. Thus, the 3-bit BCSs based shift and add unit results in fewer number of adders than the 4-bit BCSs-based shift and adder unit (reduction of four adders) with the same LD. The requirement of additional four adders would increase the complexity of the 4-bit BCSs based shift and add unit. Note that the cost of shift and add unit is independent of the number of coefficients (filter length) as the same shift and add unit is shared by all the coefficients.

The proposed 3-bit BCSs-based approach requires five 8:1 multiplexers and one 2:1 multiplexer. On the other hand, if 4-bit BCSs were used instead of 3-bit BCSs, four 16:1 multiplexers are required. Assuming an 8:1 multiplexer is equivalent to four 2:1 multiplexers and a 16:1 multiplexer is equivalent to eight 2:1 multiplexers, then the 3-bit BCS based PE requires 21 2:1 multiplexers and 4-bit BCSs-based PE requires thirty two 2:1 multiplexers, respectively. Thus, the multiplexer complexity would increase when 4-bit BCSs are used. To be more precise, for each PE with 16-bit filter coefficients, the multiplexer complexity of 4-bit BCSs-based PE is increased by eleven 2:1 multiplexers when compared to 3-bit BCSs based shift and add unit. But it can be noted that the total number of adders required for 3-bit BCS-based filter with n coefficients is $3 + 5n$ (three adders for shift and add unit and five adders for each PE) and that for 4-bit BCS-based PE is $7 + 3n$ (seven adders for shift and unit and three adders for each PE). Hence, two adders are saved for 4-bit BCS based filter for each PE. From the above discussion, it can be concluded that if 4-bit BCSs were used instead of 3-bit BCSs,

the complexity of shift and add unit and multiplexer unit of PE would have increased, whereas complexity of final adder unit would decrease. To provide a quantitative comparison, let us consider a 16-bit (W = 16) coefficient with an 8-bit quantized (x = 8) input signal. The proposed 3-bit BCSs-based CSM architecture requires twenty one 2:1 multiplexers of word length x + 2 =10 bits and 4-bit BCSs-based CSM architecture requires thirty two 2:1 multiplexers of word length x + 3 = 11 bits. For a 1-bit 2:1 multiplexer, we need eight 2-input NAND gates. This means the 3-bit BCSs-based CSM architecture requires 21 × 8 × 10 = 1680 NAND gates whereas the 4-bit BCS based CSM architecture requires 32 × 8 × 11 = 2816 NAND gates. The 3-bit BCSs-based shift and add unit requires three adders with adder-length [number of full adders (FAs)] of 10-bits each. Thus, roughly 3-bit BCSs-based shift and add unit requires 30 FAs (assuming ripple carry addition). For each FA, we require fifteen 2-input NAND gates. Thus, the 3-bit BCSs-based shift and add unit requires 30 × 15 = 450 NAND gates. Similarly the 4-bit BCSs-based shift and add unit requires approximately seven adders of adder-length 7 × 11 = 77 FAs. Thus, the 4-bit BCSs-based shift and add unit requires 77 × 15 = 1155 NAND gates. For the final adder unit, the proposed 3-bit BCS-based PE requires five adders (as shown in Fig. 4.) Adders A1 to A3 require 13 FAs {10 (output word length of shift and add unit) + 3 (shift-length)}, A4 requires 13 + 6 = 19 FAs and A5 requires 19 + 6 = 25 FAs. Thus, a total of 83 FAs are required. This means the 3-bit BCS based PE requires 83 × 15 = 1245 NAND gates. Similarly the 4-bit BCSs-based PE requires three adders. These three adders will require 15 + 15 + 15 + 8 = 53 FAs. Thus, the 4-bit BCSs-based PE requires 53 × 15 = 795 NAND gates. Now considering the total complexity of PE (Note that complexity of PE is directly proportional to the number of filter coefficients and total complexity = complexity of multiplexers + complexity of final adder unit), the 3-bit BCS based PE requires 2925 NAND gates. The 4-bit BCS-based PE requires 3611 NAND gates. Thus, for a filter with n taps, the 3-bit BCSs-based CSM architecture requires (450+2925n) NAND gates whereas the 4-bit BCSs-based CSM architecture requires (1155 + 3611n) NAND gates. Thus, it is very evident that the 3-bit BCSs-based shift and add unit results in low complexity implementation when compared to the 4-bit BCSs based implementation. Nevertheless, it must be noted that the CSM architecture can be easily modified to incorporate 4-bit or 5-bit shift and add unit based CSM architectures, if there is such a requirement. In the CSM approach, the coefficients are directly stored in the LUT and hence complete redundancy in coefficient multiplication is not avoided. Also in case of the outputs of any of the multiplexers becoming zero, the adder corresponding to that mux will be used, which is not required if the output is zero. But it can be seen that the adders at the output of the multiplexers can be combined in many ways and hence the best power solution saving can be utilized. Also carry save adders can be employed if much faster operation is required. The drawbacks in CSM are resolved by employing the BCSE algorithm proposed by us in [6]. This forms the PSM architecture which is explained in the next section.

III. ARCHITECTURE OF PSM

The PSM is based on the BCSE algorithm presented in our previous work [4]. 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 [5]. 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 coefficient that has the maximum number of non-zero operands). The architecture of PE for PSM is shown in Fig. 5. The coefficient word length 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 pass band (ωp) and stop band (ωs) frequency specifications given by 1) $\omega p = 0.1\pi$, $\omega s = 0.12\pi$; 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*= [1010011001010011]. By using the BCSE [6], substituting 2 = [1 1], 3 = [1 0 1], (11) becomes *h* = [3000020003000020]. (12) Then (12) will be stored in the LUT as 000001101011011110 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.

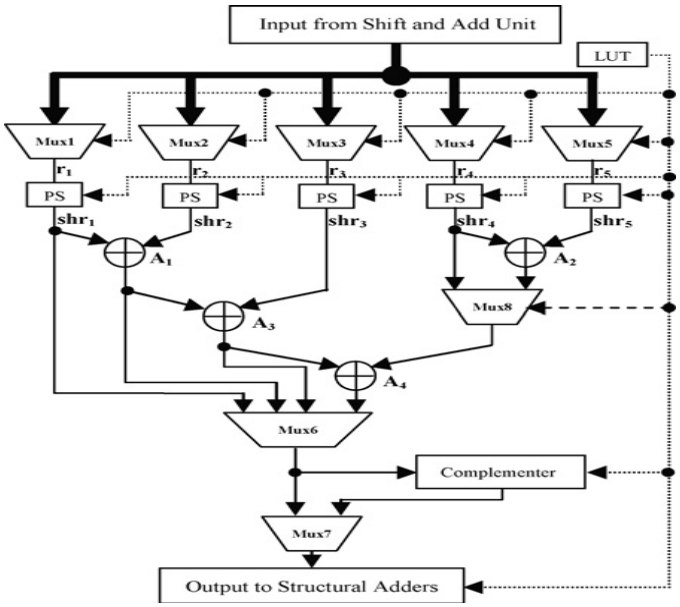

Fig 5. Architecture of PE for PSM.

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, 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; first, it guarantees a reduced number of additions compared to CSM, and second it offers the flexibility of changing the word length of coefficients. The same PSM architecture designed for 16-bit coefficients is capable of operating for any coefficient word length less than 16 bits. This means, if the word length 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 word length FIR filter architecture. This means that the coefficient word length of the proposed PSM architecture can be changed dynamically without any change in hardware.

IV. COMPARISON BETWEEN CSM AND PSM

The idea of CSM is to split the filter coefficients into groups of three bits and use these groups as selectors to multiplexer unit and obtain the product $h(x[n])$. This doesn't guarantee the minimum number of additions to be performed. In PSM, since the BCSE algorithm is employed, the number of additions to be performed will always be reduced compared to CSM. This can be illustrated as follows. Consider the coefficient *h* = [010100001010]. If CSM is employed, always four multiplexers are needed and these mean the shift and add unit in Fig. 3 needs to be used four times. Thus, always three additions are required for CSM. But if PSM is used, first we apply BCSE, then *h*1 = [020000002000]. The output computation requires only two additions, one for *h*1 and one for obtaining 2 = [101]. This reduction is significant for higher order filters. The filters used in

SDR channelizers must have a large number of taps to meet the stringent adjacent channel attenuation specifications. Therefore, the proposed PSM architecture is best suited for the channel filters in SDRs. In the case of PSM, the final shifting is done based on the values from LUT using programmable shifters whereas in the case of CSM, the shifts are constants as we are always splitting or partitioning the filter coefficients into groups of 3-bits. Thus, the CSM architecture results in faster coefficient multiplication operation at the cost of few extra adders compared to PSM architecture whereas the PSM architecture results in fewer number of additions and thus less area and power consumption compared to the CSM architecture. Another advantage of PSM is that it is independent of the word length of the filter coefficients. For PSM architecture, the number of multiplexers is fixed based on the number of BCSs present in a given coefficient set (worst case-coefficient of the set). Thus, even if the word length changes, it hardly affects the architecture of PSM. It was pointed out that for many filter taps, the highest coefficient precision is not required. Valuable hardware resources will be wasted if all taps are implemented with the highest precision. The proposed PSM can be implemented for dynamically varying coefficient precision as it is word length independent. One of the limitations of the PSM architecture is that it requires pre-analysis of filter coefficients and hence on-the-fly reconfigurability is not always feasible. But this restriction does not impose constraints on popular reconfigurable filter applications like wireless communications. This is because in such applications, we have a distinct filter for each communication standard and the coefficients of the filter are fixed for a specific standard. In other words, when the communication system is operating on a particular wireless standard, the filter coefficients do not change, i.e., the filter is not required to be an adaptive filter. When the system changes its mode of operation to a different wireless communication standard (as in the case of a multi-standard transceiver), the coefficient set corresponding to the specification of the new standard is loaded (replacing the current filter coefficients). Note that the coefficients of the new standard are known beforehand (pre-stored) and therefore the pre-analysis can be done offline and the problem with reconfigurability can be solved. In this paper, we have employed tree-structured adder for the final adder unit in both CSM and PSM architectures. But it is possible to further optimize the CSM architecture by employing techniques such as compression techniques. 3:2 or 4:2 compressors can be employed for carry free addition making the entire final adder unit more power efficient with improved speed of operation. But the use of addition structures other than tree-structured adder would impair the flexibility of PSM architecture. For example, the multiplexer Mux6 in our PSM architecture, which is employed to load/unload the adders, plays a significant role in achieving low power consumption and dynamic word length capabilities. If compression techniques were to be employed, the final adder unit should be made free of multiplexers including Mux 6, which will in turn impair above merits of PSM approach.Page Layout An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

V. CONCLUSION

We have proposed two new approaches namely, CSM and PSM, for implementing reconfigurable higher order filters with low complexity. The CSM architecture results in high speed filters and PSM architecture results in low area and thus low power filter implementations. The PSM also provides the flexibility of changing the filter coefficient word lengths dynamically. We have implemented the architectures on Virtex-II FPGA. The proposed reconfigurable architectures can be easily modified to employ any CSE (MCM) method. Thus, our method is a general approach for low complexity reconfigurable channel filters.

REFERENCES

[1]. T. Hentschel and G. Fettweis, “Software radio receivers,” in *CDMA Techniques for Third Generation Mobile Systems*. Dordrecht, The Netherlands: Kluwer Academic, 1999, pp. 257–283.

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

[3]. R. I. Hartley, “Subexpression sharing in filters using canonic signed digit multipliers,” *IEEE Trans. Circuits Syst. II*, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[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.

[5]. M. M. Peiro, E. I. Boemo, and L. Wanhammar, “Design of high-speed multiplierless filters using a nonrecursive signed common subexpression algorithm,” *IEEE Trans. Circuits Syst. II*.

.