

FPGA Implementation of Novel Reconfigurable Pipelined Architectures for Low Complexity FIR Filters



Authors:

¹ **S. Alex ***, ² **J. Selvakumar**

Address For correspondence:

¹ PG Student, Department of ECE, SRM University, Kattankulathur

² Assistant Professor, Department of ECE, SRM University, Kattankulathur

Abstract:

The Finite Impulse Response (FIR) filters require two main factors known as low complexity and reconfigurability. The FIR filters are widely applied in multistandard wireless communications. The two significant factors that are achieved by proposing two main architectures namely Constant Shift Method (CSM) and Programmable Shift Method (PSM). The major factor of filter complexity is number of adders requires in the multiplier unit. The Common Sub-expression Elimination (CSE) algorithm used to efficiently reduce the number of adders in the multiplier unit and the introduction of the Canonical Signed Digit method (CSD) to implement the low complexity higher order FIR filter is proposed in this paper. The design results shows that the proposed architectures offers a good area and power reductions compared to all other existing methods to implement FIR filters. Here compared to CSM, PSM based CSD representation for filter coefficient offers a good area and power reduction with the cost of little high delay. To reduce the delay in PSM we are going to introduce a new architecture named as pipelined PSM architecture. The pipelined PSM architecture is efficiently reduce the area and power with less delay.

Keywords: Channelizer, Common Sub-expression Elimination, FIR filter, high level synthesis, reconfigurability, Software Defined Radio, pipelined PSM.

1. Introduction

The reconfigurable FIR filters are widely used in multiband mobile communication system. The filters using in mobile communication system must be operating in low frequency and realize to consumes less power and high speed [1]. The advance technologies in mobile communication systems are demanding the low power and low complexity techniques. The Software Defined Radio (SDR) and the FIR filter researches are focused on reconfigurable realizations [2]. The SDR technology used to replace the analog signal processing with digital signal processing in order to provide flexible reconfiguration. A SDR design must meet today's reconfigurability requirements and adapt to emerging standards, as well as accommodate cost, power and performance demands. Reconfigurability of the receiver to work with different wireless communication standards is another key requirement in an SDR. Generally the complexity of FIR filter depends upon the number of adders performs in the multiplier unit. Channelizer is known as the most important block of the SDR which operates in high sampling rate but the SDR must be realizing of low power consumption and high speed. Using a bank of FIR filters in the channel filters introduces the multiple numbers of narrowband channels from a wideband signal.

alexmartyn88@gmail.com * Corresponding Author Email-Id

By replacing the channel filters by poly phase is realizing to operate in low sampling rate but the poly phase filter is reducing the speed of the operation. The co-efficient multipliers are determining the complexity of FIR filter design. The complexity of FIR filter based on the co-efficient multipliers is efficiently reduced by Binary Common Subexpression Elimination (BCSE) method. 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 [3]. Generally the large number of multiplications requirement in FIR filter which increase the area and power.

The number of multiplication operations is reduced by replacing them into addition, subtraction and shifting methods. The BCSE method was modified to eliminate the logic dept common subexpression elimination (BCSE) method which provided improved adder reductions and thus low complexity FIR filters compared 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[3]. But the pseudo floating point method is having limited channel length and this method is only suitable for where the filter co-efficient are having fixed channel length. But in the reconfigurable filter the channel length of the co-efficient filter is based on the application. Either a fully programmable multiply-accumulate (MAC) based filter processor or dedicated architectures included in this design where the filter coefficients can be stored in registers. The major components of this architecture is a data path with a single MAC unit, data and program memories, and a control unit and the data path includes a 16-bit adder/ subtractor, a multiplier, and a 32-bit accumulator.

The reconfigurability of the FIR filter is based on the Reconfigurable multiplier block(ReMB), which used to generate an all filter co-efficients and the multiplexer will select the suitable one depends on input. In the CSM architecture the filter co-efficients are consider as a constants and its stored in LUT (look up table). Here the filter co-efficients are portioned into fixed groups to eliminate the redundancy. In the PSM architecture the filter co-efficients are directly stored in LUT and which has a pre-analysis to reduce redundancy using the BCSE algorithm. The advantage of CSM architecture is it offers a high speed filters, but it requires a little more area and power compare to PSM. But the PSM offers a low power consumption

and less area efficiently at the cost of slight increase in delay. To reduce this delay we proposed new architecture named as PIPELINED PROGRAMMABLE SHIFT METHOD architecture. The pipeline is divided into segments and each segment can execute its operation concurrently with the other segments. When a segment completes an operation, it passes the result to the next segment in the pipeline and fetches the next operation from the preceding segment. A pipeline is the continuous and somewhat overlapped movement of instruction to the processor or in the arithmetic steps taken by the processor to perform an instruction. The advantage of pipelining is More efficient use of processor and quicker time of execution of large number of instructions. Here the pipelining concept is used to reduce the delay in programmable shift method architecture without affecting the area and power performance. This paper is organized as follows. The CSE method is reviewed in section II. The conventional filter architecture reviewed in Section III. In Section IV, the proposed FIR filter architecture is introduced. Design results and comparison are shown in Section V. Section VI provides the conclusion.

2. Binary Common Subexpression Elimination (BCSE) method

A CSE algorithm using binary representation of coefficients for the implementation of higher order FIR filter with a fewer number of adders than CSD-based

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} \quad (1)$$

$$\begin{aligned} Y_0 &= X_0 + X_1 + X_2 + X_3 \\ Y_1 &= X_0 L + X_1 - X_2 - X_3 L \\ Y_2 &= X_0 - X_1 - X_2 + X_3 \\ Y_3 &= X_0 - X_1 L + X_2 L - X_3 \end{aligned} \quad D_0 = X_0 + X_3$$

$$\begin{aligned} Y_0 &= D_0 + X_1 + X_2 \\ Y_1 &= X_0 L + X_1 - X_2 - X_3 L \\ Y_2 &= D_0 - X_1 - X_2 \\ Y_3 &= X_0 - X_1 L + X_2 L - X_3 \end{aligned} \quad D_1 = X_1 - X_2$$

$$\begin{aligned}
 Y_0 &= D_0 + X_1 + X_2 \\
 Y_1 &= X_0 L + D_1 - X_3 L \\
 Y_2 &= D_0 - D_1 \\
 Y_3 &= X_0 - X_1 L + X_2 L - X_3 \\
 D_2 &= X_1 + X_2 \\
 Y_0 &= D_0 + D_2 \\
 Y_1 &= X_0 L + D_1 - X_3 L \\
 Y_2 &= D_0 - D_1 \\
 Y_3 &= X_0 - D_2 L - X_3 \\
 D_3 &= X_0 - X_3 \\
 Y_0 &= D_0 + D_2 \\
 Y_1 &= D_1 - D_3 L \\
 Y_2 &= D_0 - D_1 \\
 Y_3 &= D_2 L - D_3
 \end{aligned}
 \tag{2}$$

CSE method is more efficient in reducing the number of adders needed to realize the multipliers when the filter coefficients are represented in the binary form. The observation is that the number of unpaired bits (bits that do not form Common subexpressions (CS's)) is considerably few for binary coefficients compared to CSD coefficients, particularly for higher order filters. The BCSE technique focuses on eliminating redundant computations in coefficient multipliers by reusing the most common binary bit patterns (BCSs) present in coefficients. The number of BCSs that can be formed in an n-bit binary number is $2n - (n+1)$.

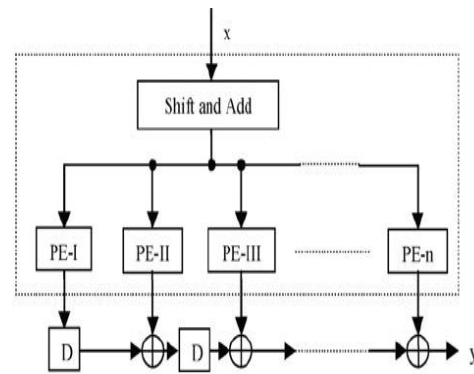
3. Filter Architectures

In this section, the proposed FIR filter architecture is presented. Fig.1 shows proposed FIR filter architecture based on the transposed direct form. The dotted portion in Fig. 3 represents the Multiplier Block (MB) [coefficient multiplier share the same input]. The MB reduces the complexity of the filter implementations, by exploiting MCM. The redundancy occurs in MCM, that redundant computations are eliminated using greedy CSE. 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. 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 ($x[n] * h$) or in a parallel way, where parallel PE architectures are employed.

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.

Fig.1. Transposed direct form of an FIR filter



3.1. The Reconfigurable CSM Architecture for Low Complexity

The CSM architecture is quite straight forward. The basic design in this approach is to store the coefficients directly in the LUT. These coefficients are divided into groups of 3-bits and are used as the select signal for the multiplexers. In this architecture the number of multiplexer units required is $\lceil n/3 \rceil$, where n is the word length of the filter coefficients. For example, if the filter coefficients are 9-bit, then the number of multiplexers required is 3. This approach can be explained with the help of a 9-bit coefficient $h = „0.111111111”$. This h is the worst-case 9-bit coefficient since all the bits are nonzero. Since $n=9$, the number of multiplexers required is 3. The coefficient h is expressed as

$$y = 2^{-1}(x + 2^{-2}x + 2^{-3}x + 2^{-4}x + 2^{-5}x + 2^{-6}x + 2^{-7}x + 2^{-8}x + 2^{-9}x)$$

By partitioning equation (8), we obtain

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

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

Now the terms $(x + 2^{-1}x + 2^{-2}x)$ and $(x + 2^{-1}x)$ can be obtained from the shift and add unit. Then by using the 3 multiplexers, precisely using two 8:1 and

one 4:1 (for the last two bits of the filter coefficients), the intermediate sums shown inside the brackets of (16) can be obtained.

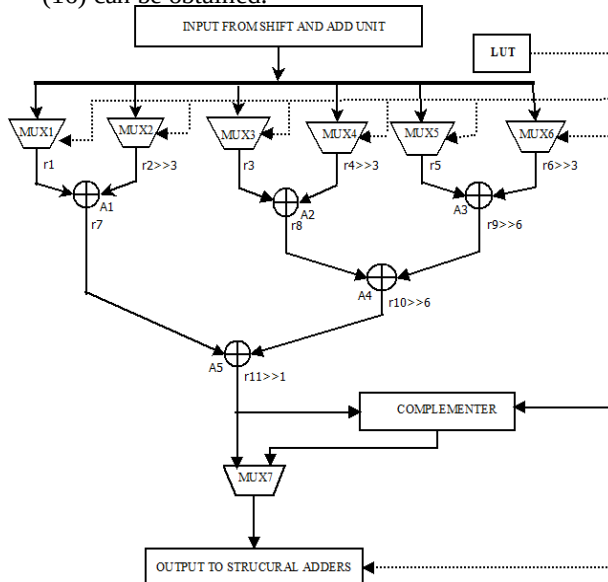


Fig.2: Architecture of PE for CSM

The final shifter unit will perform the shift operations 2-1, 2-3 and 2-6. Since these shifts are always constant, programmable shifters are not required. The final adder unit will add all the intermediate sums to obtain $h \cdot x$ [1]. The CSM architecture for the 16-bit filter coefficient is shown in Fig. 2.

The steps involved in CSM are as follows:

Step 1: Get the input x .

Step 2: Get the coefficients from the LUT and use as the select signal for the multiplexers.

Step 3: Perform the final shifting function on the output of the multiplexer.

Step 4: Perform the addition of intermediate sums using the final adder unit.

Step 5: Store the final result, $h \cdot x$, in the delay unit „D“.

Step 6: Go to step 2 if the coefficients in the LUT are not finished, else go to 1.

The three most significant bits of the coefficient will be given as the select signal to the Mux1, the next 3-bits to Mux2 and so on till the least significant bits to the last multiplexer.

3.2. The Reconfigurable PSM Architecture for Low Complexity

The PSM approach is based on the common subexpression elimination algorithm presented. Unlike the CSM method where constant shifts are used, the PSM employs programmable shifters. The advantage of PSM over CSM is that the former architecture always ensures the minimum number of additions and

thus minimum power consumption. This is because PSM has a pre analysis part. The filter coefficients are analyzed using the CSE algorithm [7]. Thus the redundant computations (additions) are eliminated and the resulting coefficients in a coded format are stored in the LUT. The coding can be explained as given below. Consider the coefficient h , $h = [1010011001010011]$ (17) By using the CSE, substituting $2 = [1 \ 1]$, $3 = [1 \ 0 \ 1]$, (16) becomes $h = [3000020003000020]$ (18) Then (12) will be stored in the LUT as $\{1, 3\}$, $\{6, 2\}$, $\{10, 3\}$, $\{15, 2\}$ which can be represented as $\{x, y\}$, where x represents the shift value and the y represents the BCS (7) to (10). The LUT contains the data in the form $\{x, y\}$. Since x can have 8 possible combinations (from [000] to [111]), it requires 3 bits, and y can have values from [0001] to [1111] for a 16-bit coefficient and hence requires 4 bits. (It must be noted that 2-1 is being applied always after final addition (17) and hence 2-16 will not occur). Thus for storing $\{x, y\}$ 7 bits are required. 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 greedy CSE. The number of multiplexers will be corresponding to the coefficient that has the maximum number of operands. The architecture for the PSM method with programmable shifts (PS) is shown in Fig. 3.

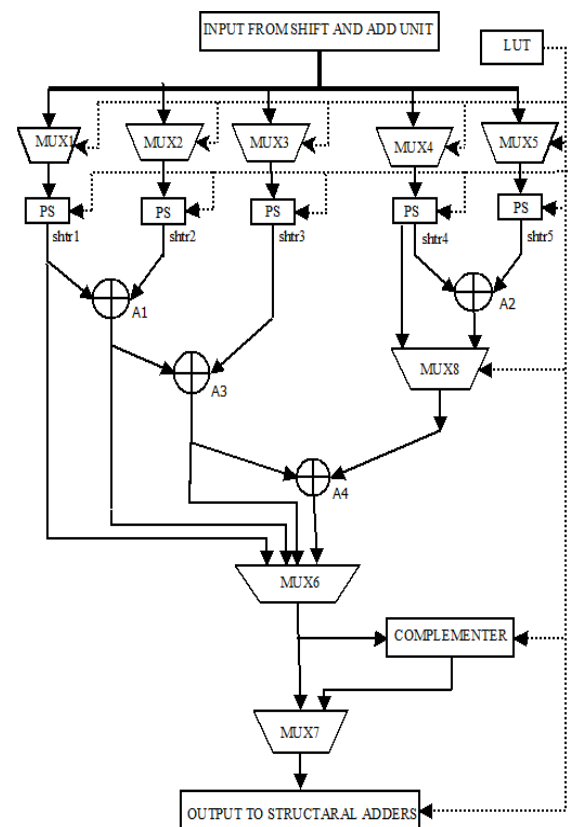


Fig.3: Architecture of PE for PSM.

The steps involved in PSM are as follows:

Step 1: Obtain the BCSs from filter coefficients using CSE algorithm.

Step 2: Store the resultant coefficients in the prescribed format as in (18) in the LUT.

Step 3: Get the input x .

Step 4: Get the coefficients from the LUT and use as the select signal for the multiplexers and the programmable shifters.

Step 5: Perform the final shifting function on the output of the multiplexer using PS. Fig. 3 PSM architecture for 16bit coefficient.

Step 6: Perform the addition of intermediate sums using the final adder unit.

Step 7: Store the final result, $h*x$, in the delay unit D'' .

Step 8: Go to step 4 if the coefficients in the LUT are not finished, else go to 3

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.

4. The proposed Programmable Shift Method Architecture.

Here we are going to propose a two new modified architectures named as programmable shift method architecture without complementer and the Pipelined programmable shift method architectures. If the filter coefficients stored in LUT are only the positive value means the complementer is not necessary. In the above Programmable shift method architecture removal of complementer would not affect the output value of PSM for positive filter coefficients. Due to removal of complementer in the above PSM architecture mentioned in Fig 3 we can reduce the mux7. This modified PSM architecture offers a efficient power reduction and area compared to normal PSM architecture for positive coefficients.

4.1. The modified PSM architecture for positive coefficients.

In the above PSM architecture mentioned in fig 3 the output of the mux6 is forwarded to mux7 and the complementer. If the output of the mux6 is having negative coefficients the complementer is used to complement the output which means the function of

the complementer is used invert the values of negative coefficients. Here the complemented values are multiplied with the original output with the help of mux7 and the final output is taken from mux7. When the output of mux6 occurs the positive value the complement of output is not necessary. But the PSM architecture mentioned in fig 3 is offering the same steps for both positive and negative coefficients. But for the positive coefficients the complementer and the mux7 are represented in the fig 3 is not necessary. So in fig 3 we removed the complementer and mux7 to provide efficient power and area management for positive coefficients. The new modified architecture is represented in fig 4

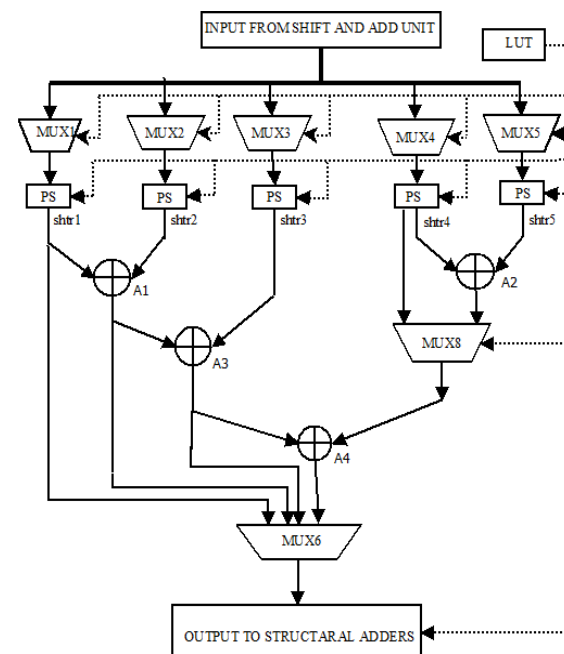
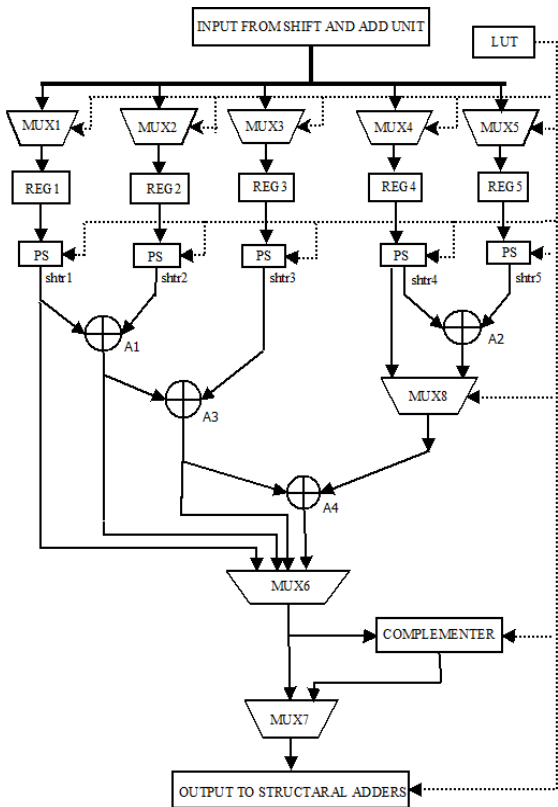


Fig 4: The PSM architecture for positive coefficients

4.2. The Proposed Pipelined Programmable Shift Method Architecture

The PSM architecture offers a good area and power reduction compare to all other reconfigurable Fir filter architecture as well as CSM. But the delay is little bit high compare to the Constant shift method architecture. In the PSM the programmable shifters are used to do the shift operations this shifters are the major factor for the delay. To overcome this delay we are going to propose a new modified PSM architecture named as pipelined Programmable shift method architecture. To reduce the sampling period or delay, we can introduce extra pipelining registers between the PS and multipliers. Then the structure of PSM is partitioned into two stages and the data produced in the first stage will be stored in the introduced registers, delaying one clock to the second stage. In

the basic PSM architecture represented in fig 3 we are going to introduce the pipelining between the programmable shifters and multipliers. The output of the multipliers is stored in the pipelining registers to produce one extra delay to the PS to reduce the critical path. The first portion of the architecture known as input, LUT and multipliers are used to multiply the inputs and the filter coefficients. The intermediate output is stored in the pipelining registers; here the registers are act as like cache memory.



The second portion of the architecture is known as the programmable shifters and the final adder unit. In the basic PSM architecture represented in the figure3, the programmable shifters are directly connected with muxes. The each output of the mux is waiting for the programmable shifters to finish its operation. The multipliers does not provide any outputs until the shifting operation is done for the past samples. Due to this operation of mux and the programmable shifters the sampling period or delay is high. But in the proposed pipelining architecture the output of the mux is stored in the temporary pipelining registers. The PS receiving the input from this temporary register and the muxes are provides the outputs for each and every clock. The pipelining registers are introducing one delay between the PS and multipliers to reduce the delay and the critical path.

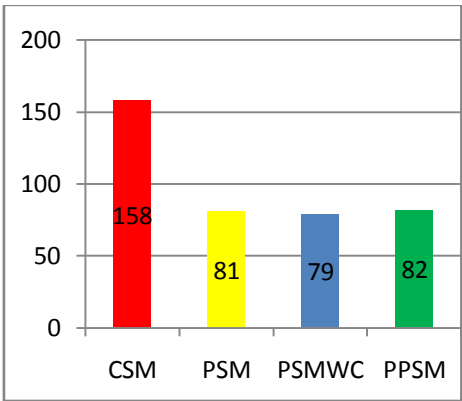
5. Results and Comparison

In this section, the synthesis results of the CSM, PSM and modified PSM for positive filter coefficients and the pipelined PSM architectures are presented and parameters like area, power and delay are compared. The comparison table states the architecture which has low power, less area and minimum delay. The Xilinx 12.3i ISE used for synthesizing purposes. Table shows the synthesis results of CSM and PSM 16-tap FIR filter that has a coefficient word length of 16 bits.

Table 1: The Power Comparison of the Synthesis Results between the Existing and Proposed Method.

Parameter	Existing Method		Proposed Method	
	CSM	PSM	PSMWC	PPSM
Power (mW)	158	81	79	82

Graph 1: Power Comparison

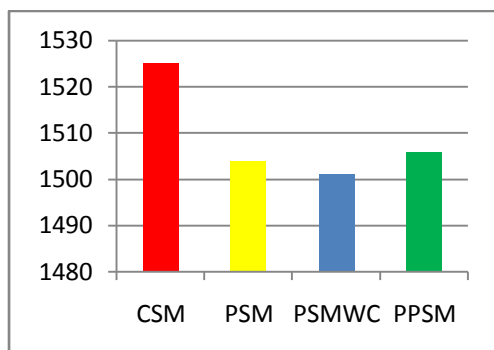


Power comparison between the existing and proposed method is represented in table 1. The PSM without complementer is offering a good power compare to all other architectures. The CSM consumes more power and the proposed method Pipelined PSM consumes little bit extra power compare to the existing one.

Table 2: The Number of Gate Count Comparison of the Synthesis Results between the Existing And Proposed Method.

Parameter	Existing Method		Proposed Method	
	CSM	PSM	PSMWC	PPSM
Number Of Gate Count	1525	1504	1501	1506

Graph 2: Number of gate count

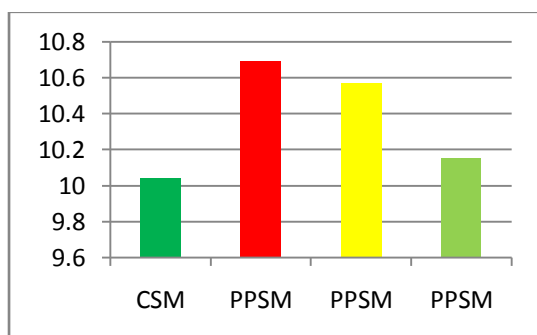


The table 2 represents the number of gates used by the existing and proposed architectures. The CSM occupies more number of gates compare to the all other architectures. The PSM uses less gates compare to the proposing method and the PSM without complements is occupied very less gates.

Table 3: The Delay Comparison Of The Synthesis Results Between The Existing And Proposed Method.

Parameter	Existing Method		Proposed Method	
	CSM	PSM	PSMWC	PPSM
Delay (nS)	10.04	10.69	10.57	10.15

Graph 3 Delay comparison



The delay comparison between the proposed and existing system states in the table3 and graph3. Here the PSM taking the more delay compare to the CSM. The proposed system pipelined PSM producing a slightly high delay compared to CSM. So here compare to all other architectures CSM exhibits very less delay.

6. Conclusion

The proposed two new approaches are 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 wordlengths dynamically. But here the proposed pipelining PSM architecture results in low area and less power filter implements over the CSM and its results a very less delay slightly higher than CSM. The proposed PSM architecture without complements results a very less power and low area over the PSM. Finally the proposed pipelined PSM resulting a low area, less power and high speed FIR filter implementations.

References:

1. Mahesh, R. and Vinod A. P. (2010) „New Reconfigurable Architectures for Implementing FIR Filters with Low Complexity“, computer-aided design of integrated circuits and systems, Vol. 29, No. 2.
2. Vinod, A.P. and Lai, E., “Low Power and High-Speed Implementation of FIR Filters for Software Defined Radio Receivers”, IEEE Transaction Wireless Communication, vol. 5, No. 7, pp. 1669–1675.
3. Mitola, J. (2000) “Object-oriented approach wireless systems engineering,” in Software Radio Architecture.
4. Wang, Y. and Mahmoodi, H. (2004) “Hardware architecture and VLSI implementation of a low-power high-performance polyphase channelizer with applications to subband adaptive filtering,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process, vol. 5., pp. 97–100.
5. Hartley, R. I. (1996) „Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers“, IEEE Trans. Circuits Syst. II, Vol. 43, No. 10, pp. 677–688.
6. Demirsoy, S. S. Kale, I. and Dempster, A. G. (2004) „Efficient Implementation of Digital Filters Using Novel Reconfigurable Multiplier Blocks“, in Proc. 38th Asilomar Conf. Signals Syst. Comput., Vol. 1. pp. 461–464.
7. Mahesh, R. and Vinod, A. P. (2006) „Reconfigurable Low Complexity FIR Filters for Software Radio Receivers“, in Proc. 17th IEEE Int. Symp. Personal Indoor Mobile Radio Commun. (PIMRC), Helsinki, Finland.
8. Potkonjak, M. Srivastava, B. and Chandrakasan, A. P. (1996) „Multiple Constant Multiplications:

Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination“, IEEE Trans. Comput.-Aided Design, Vol. 15, No. 2, pp. 151–165.

9. Mahesh, R. and Vinod, A.P. (2008) „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.

10. Vijay, s. and Vinod, A.P. (2007) „A Greedy Common Subexpression Elimination Algorithm for Implementing FIR Filters “,IEEE 1-4244-0925-7/07.

11 K. H. Chen and T. D. Chiueh, “A low-power digit-based reconfigurable FIR filter,” *IEEE Trans. Circuits Syst. II*, vol. 53, no. 8, pp. 617–621, Aug. 2006.

12 T. Zhangwen, J. Zhang, and H. Min, “A high-speed, programmable, CSD coefficient FIR filter,” *IEEE Trans. Consumer Electron.*, vol. 48, no. 4, pp. 834–837, Nov. 2002.

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 P. Tummeltshammer, J. C. Hoe, and M. Puschel, “Multiplexed multiple constant multiplication,” *IEEE Trans. Comput.-Aided Design Integr. Circuits*, vol. 26, no. 9, pp. 1551–1563, Sep. 2007.

