Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/taut20

An efficient reconfigurable FIR filter design with coefficient optimization using a modified bacterial foraging optimization algorithm

C. Kalamani, S. Lekashri, A.N. Duraivel & T. Selvin Retna Raj

To cite this article: C. Kalamani, S. Lekashri, A.N. Duraivel & T. Selvin Retna Raj (2024) An efficient reconfigurable FIR filter design with coefficient optimization using a modified bacterial foraging optimization algorithm, Automatika, 65:1, 290-303, DOI: 10.1080/00051144.2023.2296792

To link to this article: https://doi.org/10.1080/00051144.2023.2296792

AUTOMATIKA
2024, VOL. 65, NO. 1, 290–303
https://doi.org/10.1080/00051144.2023.2296792

An efficient reconfigurable FIR filter design with coefficient optimization using a modified bacterial foraging optimization algorithm

C. Kalamani[a], S. Lekashri[b], A.N. Duraivel[b] and T. Selvin Retna Raj[c]

[a]Department of Electronics and Communication Engineering, Tamilnadu College of Engineering, Coimbatore, India; [b]Department of Electronics and Communication Engineering, Kings Engineering College, Chennai, India; [c]Department of Electronics and Communication Engineering, DMI College of Engineering, Chennai, India

ABSTRACT
The digital filters play a significant role in the field of digital signal processing (DSP). The finite impulse response (FIR) filter is an attractive choice because of the ease of design and good stability. The digital filters have a wide variety of applications such as signal processing, control systems, telecommunication, etc. They are better than the analogue filters due to their performance. In recent times, software radios have achieved attention owing to requirements for integrated and reconfigurable communication systems. Hence, reconfigurations have emerged as a significant problem in the designs of FIRs. To match the frequencies of DSP applications, higher-order FIRs are required. If length of filters rises, addition and multiplication operations also increase. This paper proposes an efficient hardware design of RFIR that employs modified bacterial foraging optimizations (MBFOs) and common sub-expression eliminations (CSEs) in its executions. MBFOs output restricted counts of filter coefficients with sums of signed-power-of-two (SPT) terms while maintaining the quality of filtered responses. On obtaining coefficients, eliminations are executed by CSEs where hardware complexities are determined in terms of adders. Model sim software validated RFIRs using the Verilog code. The proposed design of RFIRs was compared with existing designs in terms of power usages, frequencies and areas.

ARTICLE HISTORY
Received 6 August 2023
Accepted 10 December 2023

KEYWORDS
Reconfigurable finite impulse response (RFIR) filter; modified bacterial foraging optimization (MBFO) algorithm; common sub-expression (CSE) elimination algorithm; transposed direct form (TDF) structure

1. Introduction

The fast-paced progress of communication technology and multi-media applications has emphasized rising demands for digital signal processing (DSP) systems [1]. Digital filters are essential systems that process discrete time signal inputs to achieve filtered outputs. The fundamental advantage of digital filters is their ability to easily alternate discrete values stored in registers and can be classified into infinite impulse responses and finite impulse responses (FIRs) where FIRs find common utility in nearly all advanced digital applications including image and speech processing, channel equalizations, signal enhancements and digital audio due to their intrinsic stable responses without poles in their transfer functions and easily attainable linear phase characteristics. FIRs have been sought for direct implementations in very large-scale integration circuits because of their inherent high stability and linear phase features [2]. Linear phase responses, required in applications such as data transmissions and crossover filters, can be achieved with symmetric coefficient FIRs. There are established filter parameters for applications including the processing of images, voices and hearing aids. Hearing aid devices must fit into a patient's ear and have reduced sizes, rapid working and batteries with longer lifetime. As a result, building FIRs that are efficient in terms of sizes, delays, power and specified specifications is critical.

Particularly, in the sphere of software-defined radio, filters based on reconfigurable architectures need to function as an integrated platform for customizing real-time digital signal processes. Moreover, these reconfigurable FIRs (RFIRs) are developed based on an adaptive strategy having variable Tapped delay line (TAP) parameters, with different frequency response demands allowed to be satisfied [3]. As mentioned earlier, RFIRs find frequent applications in places requiring low-power usage and self-adjustments. Very high-order FIRs are required to achieve great spectrum adjustments and/or noise reductions. Coefficient multiplications are critical challenges in the construction of RFIRs and their implemented designs determine performances in terms of operating frequencies, power dissipations, silicon areas and so on. Common sub-expression eliminations (CSEs) were first proposed in [4], to reduce the use of logic resources by eliminating certain repetitive computations. Among the other approaches, the CSE algorithms have been used as a

CONTACT C. Kalamani kalamec18@gmail.com Department of Electronics and Communication Engineering, Tamilnadu College of Engineering, Coimbatore, India

powerful tool for eliminating hardware redundancies and reducing area and power consumption, especially for higher-order filters in low complexity fixed-point FIR filter implementations. The main idea of CSE is to detect instances of identical bit patterns in any particular representation of the coefficients and eliminate those redundant bit patterns by reusing the results between the common sub-expressions (CSs) but with appropriate shifts in bit positions. Even though the focus was on FIRs with fixed coefficients, the CSE algorithm was examined in detail and optimized for enhancing efficiency concerning the use of logic resources. Different reconfigurable models were also built following the notions of the CSE algorithm. A pre-computer was used as a part of a paradigm called computation sharing multiplier for the production of two generic partial products. The coefficients could be multiplied simply by performing shift and adding operations on those common partial products. A Bbinary common sub-expression eliminations (BCSEs) approach was developed to reduce the number of adders that perform recurrent BCSs [5]. Two BCSE designs were investigated: constant shifts methods and programmable shifts methods. The pre-computers of these two architectures use BCSEs for the creation of required partial products, which are distributed to every processing element (PE) so that the multiplication of coefficients can be completed.

Besides its expression in the binary form, the representation of the filter coefficients can also be equivalently done in canonic signed digits (CSDs) [6]. It has been found that the CSE method earlier studied has to be much more efficient in binary compared to CSD forms. This is primarily because CSD representations need an additional sign bit, making the computational overhead increase. However, the coefficients of filters can be modified for specific outputs. As a result, effective filters may be created by obtaining new sets of coefficients and optimizing the filter's orders and word lengths [7]. Shifts and adder circuits replace the multipliers used in FIRs, and these adders are dependent on the number of SPT terms used in filter coefficients. As a result, the goal of the suggested approach was to get a set of filter coefficients with restricted SPT terms and achieve ideal word lengths and orders of filters, such that the filters would be enhanced in terms of latencies [8], area and power. Several algorithms are available to decrease the number of SPT terms.

Few methods consider how the FIR design parameters behave statistically to simplify the computation process [9]. According to previous research, a binary form of representation must be chosen for pre-computing canonic forms. This is because the occurrence frequencies of the CS patterns chosen in the former are higher compared to the latter [10]. But, as it was seen, this possibility may not be true at all times, particularly in the scenarios where the CS patterns are specified in a particular way aiming at the computational operations' reduction.

Recently, some scholars have reviewed the use of the evolutionary algorithms to design filters, and pointed out that the evolutionary algorithms are very suitable for filter design [11]. These algorithms include simulated annealing (SA) algorithm [12], genetic algorithm (GA) [13], particle swarm optimization (PSO) algorithm, differential evolution (DE) algorithm [14], ant colony optimization (ACO) algorithm [15], cuckoo search algorithm (CSA) [16], etc. These optimization algorithms have been used for digital filter design, and have made some progress. However, these also have various limitations. The SA algorithm has slow convergence, long execution time and sensitive parameters, which make it an inefficient and even infeasible algorithm. GA and ACO algorithms are difficult to apply to practical applications because of their complex structure and slow operation speed. The PSO algorithm is easy to fall into the local optimum because each particle in the swarm only searches in a limited sample space. Similar to the commonly used evolutionary algorithm, the DE algorithm achieves the optimal solution crossover and selection of the difference vector between the individuals. Therefore, for some complex optimization problems, the DE algorithm also has a local optimum and premature convergence. There are some problems in CSA, such as low convergence accuracy and low convergence speed. Furthermore, the search probability and search step have a great influence on the performance of the CSA. In other words, these mature intelligent optimization algorithms have more parameters to be adjusted, and improper parameter adjustment is easy to make the algorithm fall into local extremism, and so forth also get good filter design results. Therefore, it is worth studying to find an intelligent computational algorithm with a simple structure, strong robustness, few parameters and ease of adjustment for the design of the FIR filter.

In this work, a hardware-efficient RFIR filter design method is presented which employs modified bacterial foraging optimizations (MBFOs) and CSEs. The primary contributions made by this article are briefly described below:

- A new CSD coefficient grouping technique is introduced that only identifies if there is a single "zero" bit between two "nonzero" bits, analysing the distributions of the selected CSs amongst multiple CSD coefficients drawn from a wide range of FIR designs.
- A novel PE form for the medium-grain array architecture utilizing RFIRs is advocated to achieve optimal use of logic resources and outstanding performance.

- Using the MBFOs approach, a set of filter coefficients with a restricted amount of SPT terms is identified initially, with the quality of the filter response remaining unaffected.
- Once coefficients are obtained, CSEs are used and the hardware complexity is decided in terms of adders.
- The cascading of the proposed PE can be readily done on a module level and can be easily used for field-programmable gate arrays (FPGA) or application-specified integrated circuit (ASIC) implementations. The filters were developed by applying MBFOs for different filter orders, and the implementation of the same was done in the TDF structure.

The other sections of the technical work are structured as follows. Section 2 provides an overview of the related eeconfigurable FIR designs along with their coefficient optimization. Section 3 discusses the process of the proposed RFIR design along with the coefficient optimization utilizing the proposed modified bacterial foraging optimization algorithm (MBFOA) with CSE-based concepts. In Section 4, the results are studied. Section 5 provides the conclusion and the work intended for the future.

2. Related work

The designs to implement the FIRs can be done in many ways for both fixed and RFIR applications where the latter filter coefficients can be varied dynamically during run time. Meanwhile, the fixed architecture has pre-defined coefficients. The techniques studied are primarily different in terms of their filter structure. Few methods choose block-based mechanisms owing to their rapid speed in parallel processing.

Mohanty and Meher [17] proposed powerful distributed arithmetic (DA) derivations in their implementations of block least mean squares (BLMSs). To compute filter outputs and weight increments of BLMSs, their suggested DA-based architecture leveraged an innovative look-up table (LUT) sharing technique. A parallel architecture executed BLMSs in compliance with suggested DA derivations and adaptive digital filters (ADFs). When compared to the finest DA-based least mean square (LMS) models available, their suggested schemes had around L/6 time's adders and LUT words with throughputs closer to L times of the balance. This was the most significant advantage of their suggested design since area delay products (ADPs) decreased, particularly during implementations of high-order ADFs with bigger block sizes. It was shown from ASIC synthesis results that the study's model with a filter length of 64 exhibited 14–30% reductions in ADPs and 25–37% reductions in Energy per operation compared with other architectures using block sizes of 4 and 8.

Guo and DeBrunner [18] filtered finite impulse responses adaptively using DA-based techniques, unlike traditional DA approaches where LUTs with delayed and scaled inputs are accessed with coefficients. The study created two outstanding LUT updating strategies used LMSs to ensure that weights were updated and mean square errors (MSEs) between the predicted and needed outputs were reduced. Their results showed that their designs enhanced speeds, minimized calculation complexity and reduced area costs.

Gunasekaran and Manikandan [19] proposed architectural frameworks for creating RFIR filters with decreased power usage and space optimizations. FIR digital filters are used in DSP because of their linear phases, low finite precision errors, stability and outstanding implementations. Their suggested structures accomplished low power and area by modifying adders at appropriate locations and thus reducing power consumption and area. Their suggested structure outperformed other available RFIRs when simulated and validated on Spartan-3 xc3s200-5pq208 FPGAs.

Haridas and George [20] used area-optimized architectures of low-power FIRs based on spanning trees and adapted Booth multipliers for direct form implementations. Area-optimized modified spanning tree adders were presented to increase the area efficacies of FIRs. Their design was implemented using Xilinx 14.2 ISE tools, Model Sim and VHDL (hardware description language) programming. FIRs were constructed in the MATLAB Simulink tool where multiple filter coefficients were selected.

Ramachary and Siva [21] introduced novel pipelined structures for low-power, high-throughput and low-area implementations of adaptive filters based on DA. Their proposed technique achieved a significant boost in throughput rates by updating parallel LUTs, parallel implementations of filters and weight updates. Traditional adder-based shift accumulations for DA-based inner product calculations were replaced by conditional signed carry-save accumulations to reduce sampling periods and area complexities [22]. The study suggested architecture reduced power dissipations using fast-bit clocks to collect carry-save adders while using slower clocks in other operations. The study's comparisons with existing DA-based architectures [23] showed an equivalent number of multiplexors but with smaller LUTs and only half of the adders' count. Their synthesis showed that their proposed design's power consumption was 13% lower while ADPs were 29% lower than previous DA-based adaptive filters with an average of filters with lengths of 16 and 32. When compared to outstanding designs, the suggested structure generated 9.5 times less power and 4.6 times reductions in ADPs.

Kadam et al. [24] examined the pipeline and parallel processing structures of FIRs for efficiently implementing FPGAs. It was proven from the simulation outcomes that the efficiency of the parallel processing structure is much better than that of the pipeline structure. Also, a fast FIR structure is highly desirable in comparison with a traditional parallel processing structure as its hardware complexity is reduced. There is a 25% improvement in an area in fast FIR structures when matched with traditional parallel processing structures for similar performance. Pristach et al. [25] presented an improved structure of digital infrared filters. For data storage, their proposed structures had singular units for accumulations of multiplication units using random access memories. Area requirements were reduced by serial computations. The proposed structure was suitable for application-specific integrated circuits and FPGAs. The structure's key advantages are increased operational frequency, decreased power dissipation and less area utilization in specified conditions.

Jia et al. [26] suggested a simple CSD coefficient group for reducing the count of CSs and in addition, the occurrence of CSs was statistically examined for various types of FIRs, and representations of distributions for likely CS patterns in generated 16-bit coefficients. Their plans resulted in a novel processing design that produced medium-grain arrays that were effective for computational implementations of FIR reconfigurations. It is shown from the experimental results that these design implementations usually gain a 21% reduced silicon area, power dissipation reduced by 20% and a 14% increase in operational speed compared to other traditional FIR models.

Sriadibhatla and Baboji [27] present the design and implementation of an area and power-efficient RFIR filter. We present a method for designing are configurable filter with low binary complexity (BC) coefficients and thus optimize the filter while satisfying the design specifications. The total number of nonzero binary bits is taken as a measure of the BC of a coefficient. We propose two implementation architectures, namely signed-magnitude architecture (SMA) and signed-decimal architecture (SDA), which are based on the 3-bit BCSE algorithm and vertical horizontal BCSE algorithm, respectively. SMA and SDA reduce the redundant computations of the coefficient multiplications in the filter. The proposed filters are synthesized on the tsmc 65 nm Complementary metal oxide semiconductor technology.

Lian and Tian [28], proposed an FIR digital filter design method based on an improved artificial bee colony (ABC) algorithm to optimize the design of the FIR filter. This improved ABC algorithm can adaptively adjust the step size of the selected neighbourhood of the nectar source location. At the same time, the information on the global optimal solution is used to guide the search for a candidate solution, which improves the global search ability of the algorithm. The improved ABC algorithm can balance the conflict between local search ability and global search ability, so it can achieve a better optimization effect. The time and space complexity of the algorithm is analysed in detail. Then, the improved ABC algorithm is used to design low-pass, band-pass and band-stop filters.

From the above discussion, it is identified that each method has some advantages and disadvantages. To meet the correct frequency requirements in many DSP applications, higher-order FIRs are required. However, when the filter length increases, the number of adds and multiplications increases linearly, increasing the computing complexity. The existing method exhibits a slight rise in the filter complexity, but the frequency response is considerably improved.

3. Proposed methodology

Digital filters are costly in terms of hardware and power-dissipating components in signal-processing devices. Therefore, it is always desirable to use a hardware-optimized and low-power digital filter even if the time taken for the design process is high. In this work, a hardware-optimized RFIR filter design that employs MBFOA and a CSE elimination algorithm is presented.

- A novel CSD coefficient grouping approach that only determines if singular "zero" bits are added between two "nonzero" bits, analysing selected CS distributions amongst CSD coefficients belonging to multiple FIR designs.
- A novel PE form for medium-grained array architectures utilizing RFIRs is advocated to achieve optimum logic resource use and greater performance.
- MBFOs are used to find filter coefficient sets with a restricted number of SPT terms while maintaining the quality of filter responses.
- On obtaining the coefficients, CSE eliminations are used and hardware complexity is determined in terms of adders.
- The cascading of the proposed PE can be readily done on a module level and can be easily used for FPGA or ASIC implementations. The filters were developed applying MBFO for different filter orders, and the implementation of the same was done in transposed direct form (TDF) structures.

The filters were developed with the help of MBFO-CSE for different filter orders, and the implementation of the same was done in the TDF method.

3.1. *Design of RFIRs*

FIR digital filters are being employed extensively in DSP applications as they are finite without feedback. N-tap FIRs include adders (N) and multipliers ($N + 1$)

consuming larger areas and voluminous power dissipations. FIR's transfer function can be depicted as

$$y(n) = \sum_{k=0}^{M-1} h(k) \times (n-k) \tag{1}$$

Generally, the representation of the FIRs takes the TDF, the popular structure for representing the filters. Filters are called reconfigurable if their TAP parameters are variable totally. Multipliers have an important part to play in filter design realization. The two most common filter designs are combinational and sequential multipliers. When input word lengths increase, the area required for combinational multiplications grows exponentially while space is saved when using sequential multipliers. However, its main drawback lies in its clock cycles in executions, making it complex to be applied to real-world applications. As an increased speed is necessary for filtering, its implementation is done with the help of a combinational multiplier design whose power dissipation is excessive due to a bigger area. The implementation of coefficient multiplication decides the design performance including power dissipation, operational frequency and area. Also, a popular pre-computer is designed to generate a few partial products, which are generally developed during the multiplication step. The remaining partial products developed during the multiplication process are just acquired through the shift and added operation of the pre-computed values.

When the computation $y[n]$ is performed directly, many partial products get generated and they require higher-order multiplications which in turn consume more space while increasing computing times. To reduce computational overhead, the grouping of coefficients, depicted in Figure 1, follows the procedures listed below:

i) On the existence of a 0 bit between two non-0 bits, the 3 bits are merged into one group.
ii) If multiple 0 bits exist between two non-0 bits, they are treated as two separate groups.

The coefficients are organized into three bits, starting with the least significant bit values. Figure 2 depicts the grouping flow for CSD coefficients. When subgroups are multiplied by an input X, it results in one of the alternatives, namely 1X, 2X, 3X, 4X, 5X, 6X and 7X. Pre-computations of 1X, 2X and 4X additionally simplify multiplications where the remaining numbers are determined by adding them to these pre-computed values. For example, combining 2X with 1X results in 3X, and combinations of 1X, 2X and 4X result in 7X. Thus, parallel multiplications of subgroups are

$$h(n) = \left\{ 1 \ \boxed{101} \ \boxed{001} \ \boxed{110} \ \boxed{111} \ \boxed{010} \right\}$$

Figure 1. CSD grouping method.

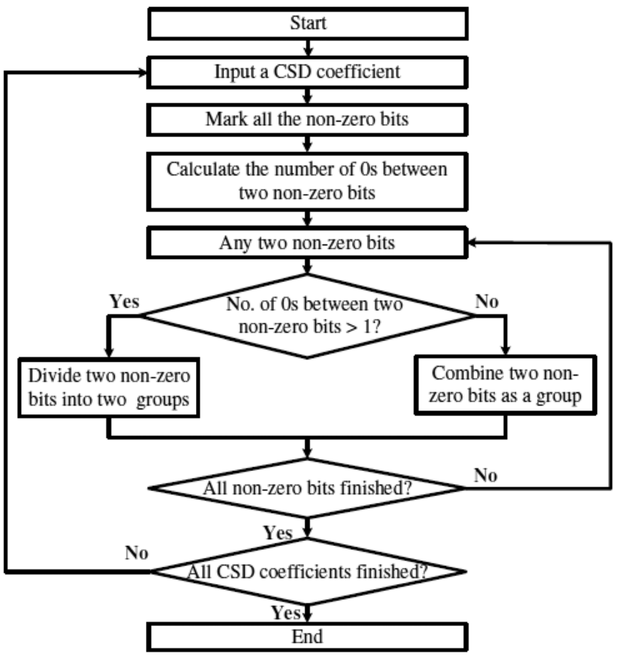

Start
Input a CSD coefficient
Mark all the non-zero bits
Calculate the number of 0s between two non-zero bits
Any two non-zero bits
Yes
No. of 0s between two non-zero bits > 1?
No
Divide two non-zero bits into two groups
Combine two non-zero bits as a group
All non-zero bits finished?
No
Yes
No
All CSD coefficients finished?
Yes
End

Figure 2. Proposed grouping flow for CSD coefficients.

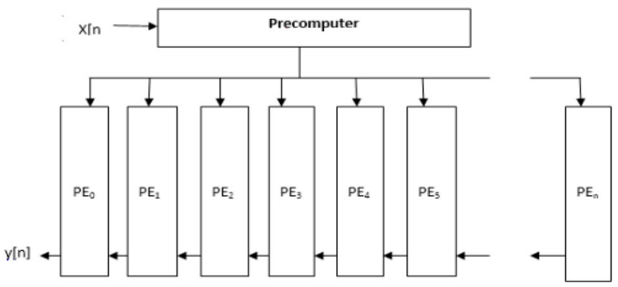


Figure 3. The architecture of RFIRs.

executed and calculated values are shifted and summarized for final results.

As seen in Figure 3, filter constructions necessitate the use of pre-computing for the three partial product computations used in coefficient multiplications along with a cascaded array of PEs. PEs conduct input multiplications in two subgroups, and in the last step, appropriate shifts are applied to all values which are then summarized for the final multiplication result.

As seen in Figure 4, Z1, Z2 and Z3 (partial products) are passed into PEs when coefficient multiplications using shifts and adds are executed. PEs have four inputs while their outputs are singular where three derived inputs are partial products while the fourth is an additional input derived from the previous PE block's output that can be used for cascading.

3.2. Problem formulation

A RFIR coefficient defined, as a summation of sums of signed-power-of-two (SPT) terms, is expressed by

$$h_{spt}(n) = \sum_{m=1}^{M} s_m, \, 2^{-m} \tag{2}$$

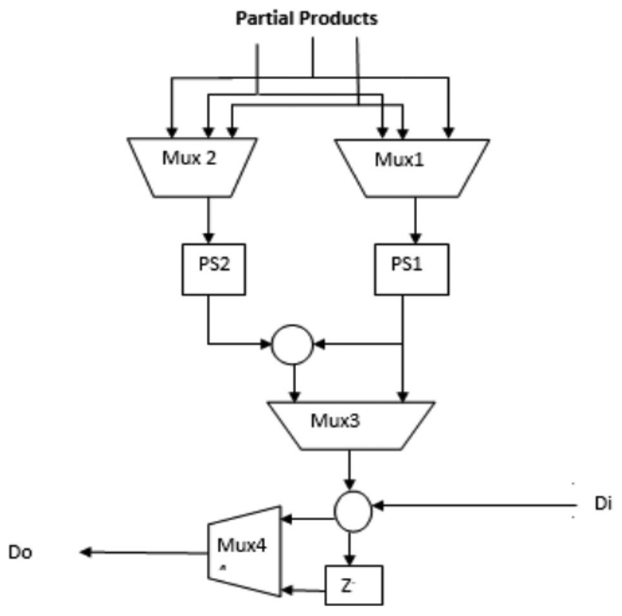


Figure 4. Operation of a single PE.

where $s_{m,n} \in \{-1, 0, 1\}$ and $n = l, \ldots .N$. N refers to lengths of filters and M indicates filter coefficient word lengths. During the design of this filter, it is necessary to decide the values of $S_{m,n}$ so that their corresponding frequency responses from $H_{spt}(\omega)$ filters are minimal normalized peak rippled (NPR) or have minimal deviations from frequency responses $H_d(\omega)$. For making Equation (2), every $s_{m,n}$ term to match differences between two non-negative terms in Equation (3) may be used:

$$h_{spt}(n) = \sum_{m=1}^{M} (q_{m,n}2^{-m} - q_{-m,n}2^{-m}) \tag{3}$$

where $q_{m,n}$ represents FIR coefficient's positive SPT terms and $q_{-m,n} \in \{0,1\}$ represents FIR coefficient's negative SPT terms. $Q_k \equiv (q_{1,1}, q_{1,2}, \ldots q_{M,N}, q_{-1,1}, q_{-1,2}, \ldots q_{-M,N})$ represents the feasible coefficient solution for k non-0 element's filters or $\sum_{n=1}^{N} \sum_{m=1}^{M} (q_{m,n} + q_{-m,n}) = k$. In this manner, a filter representation with two coefficients $h_{spt}(1) = 2^{-1} - 2^{-4}, h_{spt}(2) = 2^{-3}$, for instance, can be given by Q3 having $q_{1,1} = q_{-4,1} = q_{3,2} = 1$ while others are equal to zero $(q_{m,n})$. Assuming $\{Q_k\}$ represents a set of $(2MN/k)$ feasible solutions and $npr(Q_k)$ points to $Q_k^{\prime}s$ PR filter coefficients, $\{Q_k\}^+$ represents $\{Q_k\}$'s subset elements constituting *2MN* Q_k for the best NPR values and $Q^+_{k_{m,n}}$ stands for the elements in $\{Q_k\}^+$. The design of filters with predefined K SPT terms can be derived as appropriate Q_k that reduces NPR in filters:

$$\{npr(Q_k)\} \tag{4}$$

For the filter design problem studied, the above expression implies

$$\{npr(Q_k)\} = \{npr(q_{m,n}) + npr(Q_{k-1}|_{q_{m,n\neq 1}})\} \tag{5}$$

Dynamic programming is a bottom-up technique that begins by finding solutions to the smallest subproblems and subsequently combines these solutions incrementally till actual solutions are found. Hence, dynamic programming develops filters initially from exactly k – 1 SPT terms and minimal NPRs $\min\{npr(Q_{k-1})\}$ (the second term of Equation (5)'s right-hand side). Subsequently,new filter sets with k SPT terms $\min\{npr(Q_k)\}$ (Equation (5)'s left-hand side) are designed by adding correct SPT terms to the previous solution sets. These operations are repeated until the required SPT period counts are attained. Unfortunately, the additive connection in (5) is inaccurate because the amount of NPR enhancement that an SPT term provides is also reliant on the other SPT terms that a filter contains. As a result, the solution obtained using the standard dynamic programming technique cannot be guaranteed to be optimum.

Nevertheless, k SPT terms-based filters can be obtained by adding correct SPT terms to good filters with $k-1$ SPT terms and filter designs with SPT coefficients can be achieved by recursive optimizations similar to dynamic programming as given below:

$$\{npr(Q_k)\} \cong Q_k \in \{\{Q_{k-1}\} + \{npr(Q_k)\}\} \tag{6}$$

Recursive optimizations of (5) are proposed by using MBFOs to optimize the coefficients of RFIRs. The "Linear phase" stands for circumstances when filter responses are provided by straight line functions (linear frequency) except when the phase wraps at +/− 180 degrees. RFIRs adhere to a linear phase when their coefficients are symmetrical around the centre coefficients which are the primary coefficients equal to the final coefficient and the second coefficient is equivalent to the next-to-end coefficient. The accurate control of the various frequency spectrums is the main objective function for the designing of ideal digital filters which exhibit high nonlinearity, non-uniformity, non-differentiable and multimodal behaviour. These objective functions cannot be optimized by classical optimizations and fail to converge for global minimum solutions. To minimize all these disadvantages which are faced with the conventional optimization approaches, many researchers have used heuristic and meta-heuristic evolutionary optimizations related to evolutionary and natural techniques.

The escalating levels of power dissipation are the second major concern associated with the development of processing power and the complexity of signal processing algorithms. Two types of power dissipation occur: static dissipation, which occurs as a result of leakage current or another current from the power source, and dynamic dissipation, which occurs as a result of switching transient current. The main activities of the RFIRs have filtered coefficient multiplications and accumulations of digital inputs and attained the use of more adders and multipliers matching filter coefficient counts. Because multipliers consume a lot of power and space, it is usual to employ multiplierless realization, which replaces multipliers with adders and shift registers. The purpose of the optimization approach is to decrease the discrepancy between the obtained and necessary frequency magnitude response and adjust the filter coefficients. However, one disadvantage is that one or more design parameters, such as filter length, number of SPoT terms and filter taps, are set throughout the design process. As a result of this disadvantage, filter design becomes more sophisticated than necessary. The main aim of this study is to investigate linear phase optimization approaches using MBFOs and hardware complexity using CSEs.

3.3. Co-efficient optimizations using MBFOs

An improved technique for discrete optimizations of RFIR digital filter coefficients, whose representation is in the form of CSD codes [29], i.e. integers that may be expressed as sums or differences of powers-of-two, is introduced in this paper. To compensate for the severely non-uniform behaviour of CSD coefficient distributions, the suggested search method gives an extra nonzero digit in CSD codes to the larger coefficients. The filter complexity increases somewhat, but the improvement in frequency response is significant. The CSEs are provided in the next section to prevent this difficulty.

When all of the filter coefficients are multiplied by fixed scales, the form of frequency responses of RFIRs remains unchanged. The scale factor simply adds gains or attenuations to response frequencies. However, this scale factor also has significant influences on coefficient optimizations when the coefficients are CSDs. The reason for this improvement is that the set of numbers that can be represented by CSD codes using a constant number of nonzero digits has a highly non-uniform distribution, and thus correct scaling of the optimal filter coefficients, before rounding them to the closest CSD codes, can generally result in a significant reduction in the magnitudes of the coefficient quantization errors and it is immediately depicted in a higher frequency response. The suggested modification to the MBFOs comprises the addition of a nonzero digit to the CSD coefficient representation for the filter coefficients with large values, which results in a significant gain in performance in the scaling section of the optimization method.

3.3.1. Distribution of CSD coefficients

Representing a number in the form of sums and differences of powers-of-two is technically referred to as a radix-2 signed-digit code. The radix-2 signed-digit representation of a fractional number x is generally expressed as

$$x = \sum_{k=1}^{L} s_k 2^{-P_k} \tag{7}$$

where $s_k \in \{-1, 0, 1\}$ and $p_k \in \{0, 1, \ldots, M\}$ The representation in (7) contains $M + 1$ total (ternary) digit and L nonzero digits. Typically, a given number has many signed-digit representations. The minimum representation denotes a coding that requires a small amount of nonzero digits, from which several options may be accessible. Representations of CSDs are defined as minimum representations where there are no consecutive two nonzero digits s_k, the representations of numbers differ. They are simple procedures that translate binary representations into their CSD forms. The adder/subtractor counts for implementations of CSD coefficients are lesser than the code's non-zero digits. The main advantage of CSDs compared to typical radix-2 binary codes like 2's complement lies in the added flexibility of negative digits that allow representations of most integers with minimal nonzero digits.

The optimal distribution of realizable filter coefficient minimizations of greatest quantization errors is uniform. 2's complement can be an example of uniformly distributed representations. As seen in Figure 5, CSD coefficient distributions represented as nonzero 6-digit and 8-digit CSD codes are highly non-uniform. Gaps in 1/8 widths can be seen in the set of integers represented in Figure 3. Moreover, nonzero digit counts of CSD codes are kept constantly at 2, gaps will not decrease even if CSD code word lengths come closer to infinity (i.e. $L = 2$ and M + CO in (7)) and hence only incrementing nonzero digit counts in CSD codes can reduce these gaps.

However, incrementing nonzero digits in CSD codes may not be an effective way of reducing coefficient quantization errors. CSD coefficient distributions (refer to Figure 5) are dense for lower coefficient values. Hence, when filter coefficients are rounded to the nearest CSD codes, quantization error levels will increase for coefficients with large values. This work minimizes coefficient quantization errors without increasing their complexity (normalizing coefficient's impulse responses to a maximum value of 1):

Coefficient's impulse responses $> 1/2$ are allocated one extra nonzero digit in CSD representations.

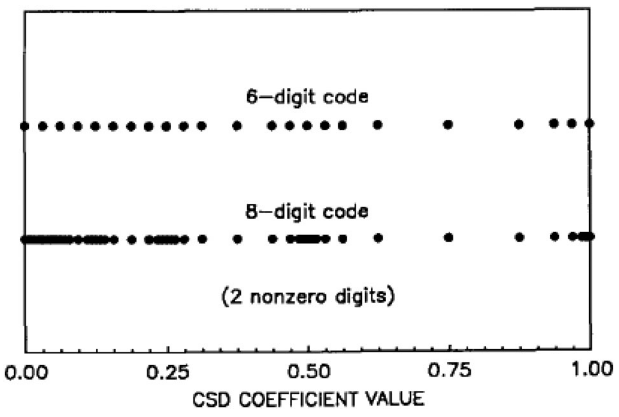


Figure 5. CSD coefficient distributions for sets of 6- and 8-digit codes having two non-zero digits.

The maximum number of RFIRs results in low hardware complexity as low-pass impulse responses generally contain a $\sin(x)/x$ envelope and thus only a small number of coefficients in the primary lobe of the $\sin(x)/x$ response will be $> 1/2$.

3.3.2. Scaling mechanism

The coefficient's impulse responses were the first to be scaled for obtaining 1 as the value of the largest coefficient. During optimizations, L and M parameters in (7) are selected for the construction of viable CSD coefficients in the range [0,1] for future use. It is worth noting that L was increased by 1 for coefficients with magnitudes greater than 1/2.

As the quantization of coefficients is non-linear, early predictions of scale factors would provide highly unrealistic results and hence scale factors need to be obtained using brute force search. Scaling by a factor of two had no influence on quantization processes, hence one octave of scale factors was explored and coefficients of filters were rounded to the nearest CSD number in the table for scale factors in the range [0.5,1] and the resulting frequency response peak weighted ripples denoted below were determined:

$$\delta = \left[\frac{\delta_p}{w, \delta_s}\right] / b \tag{8}$$

where δ stands for the pass and stop band ripple amplitude, b indicates the average pass band gains and W *is* the ripple weighing factor. Using Fast fourier transform (FFT) applications, frequency responses are computed by padding zeros to the coefficient's impulsive response. The selected FFT points were a power-of-two multiple of 8 times the filter length and step size A between scale factors was set for balancing the length of searches and quality of eventual results. Another reason for selecting scale factors was to make the most of the relationships between coefficient quantization mistakes and frequency response errors where scale factors with the lowest coefficient quantization error values were selected.

3.3.3. Modified bacterial foraging optimization algorithms

MBFOAs are based on BFOAs [30] and are intended to discover solutions to constrained numerical optimization problems (CNOPs). One of the advantages of using this method is that it eliminates the need to select the filter order beforehand. It decreases MSEs between the preferred frequency responses and the frequency responses corresponding to approximations. If a mistake occurs, the algorithm increases the filter order. All of its elements are explained in the order they are supplied.

(i) A bacterium: It implies the best solution to CNOPs (n-dimensional real value vectors identified as $x^{\rightarrow}$), represented as θi (j, G), where j is the chemotaxis loop index while G is the generation cycle's loop index. A cycle performs three inner processes, namely chemotaxis, reproductions and eliminations where swarming processes are a part of chemotaxis processes.

(ii) Chemotaxis: In this process, the bacterium of current swarms' takes tumble-twirl moves where tumbles (as studied by Passino) include search directions (i) which are produced randomly after uniform distributions and expressed as

$$\varphi(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}} \tag{9}$$

where $\Delta(i)$ is the randomly produced n-dimension real value vector from uniform distributions in the range $[-1, 1]$. Twirls permit bacteria $\theta^i(j, G)$ to stick to their search directions and travel to new positions $\theta^i(j+1, G)$. Twirl can be computed using Equation (10):

$$\theta^i(j+1, G) = \theta^i(j, G) + C(i)\varphi(i) \tag{10}$$

where (i) stands for the step size vectors and is computed by using threshold values of design variables k, as depicted in Equation (11):

$$C(i)_k = R * \left(\frac{\Delta x_k}{\sqrt{n}}\right), \quad k = 1, \ldots n \tag{11}$$

where Δx_k represents differences between the upper and lower thresholds of variables x_k: $U_k - L_k$, n implies variables to count, and $R \in [0, 1]$ is a user-determined parameter used in bacteria's step size scales. It has a constant value during searches. New positions, $\theta i\,(j + 1, G)$, may be better than prior positions $\theta^i(j, G)$ (9) or the objective function's new position may be better (10) or new positions may be possible when earlier (11) both positions are impossible but depict the reduced sum of constraint infringements. Another twirl in the same direction is executed making it the start position on completion of entries, the procedure is terminated.

(iii) Swarming: If no bacteria exist in current swarms, MBFOAs use attractor movements inside chemotaxis processes for the bacterium in swarms to move towards the bacterium in search space's most potential area, which is a practical bacterium with the best objective function values or bacterium with the least sum of constraint infringements. Feasible movements used in chemotaxis processes are shown below:

$$\theta^i(j+1, G) = \theta^i(j, G) + \beta(\theta^B(G) - \theta^i(j, G)) \tag{12}$$

where $\theta^i(j+1, G)$ stands for bacterium i's new location, while $\theta^i(j, G)$ represents its current position. (G) refers to the optimal bacterium's current position in the swarm at cycle G, and β is a parameter (user-defined) indicating the proximity of bacterium i's new position concerning the optimal bacterium (G)'s location. Attractor movements are used a single time in chemotaxis loops, while tumble-twirl movements are used in subsequent phases. Outside of their bounds, the tumble-twirl and swarming movements may both produce a variety of values. As a result, the broken limit is deleted and the violated amount is multiplied by two (i.e. $x_{\text{valid}} = 2_{*}$violated limit$- x_{\text{invalid}}$).

(iv) Reproduction: The swarm is sorted using the same three principles that are used in the chemotaxis process, and the first S_r reproduced (these bacteria are considered the best), while the rest of the $S_b - S_r$ (the worst bacteria) are discarded ($S_b$indicates the swarm size).

(v) Elimination-dispersal: This method removes the worst bacterium (j, G) based on previously established feasibility rules, and another randomly generated microbe replaces the bacterium.

Tandem-twirl MBFOA (TT-MBFOA)

TT-MBFOA revisits the tandem-twirl presented in MBFOA to increase its search potential and simplify it. Tandem-twirl is employed inside the chemotaxis process, as illustrated in this study effort in this manner. The purpose of the first twirls is to complement swarming operators by allowing the bacterium to search in additional fields of search spaces and the direction of randomly selected bacteria. Second spins use real twirls but with minuscule step size values, concentrating on minute bacterial motions. Each of the tandem-twirls proposed is described in detail below.

(1) Exploration twirl

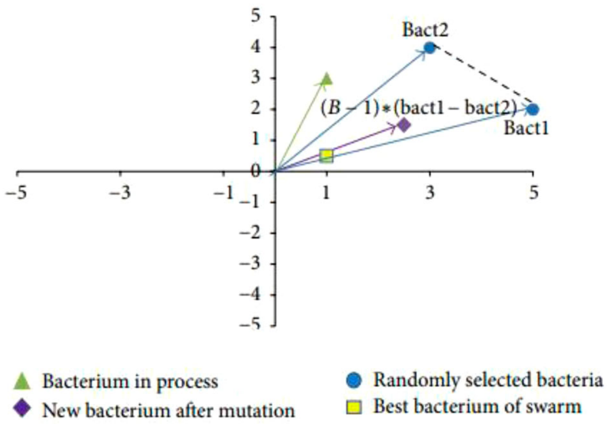
The first twirl is computed, as indicated in the following:

$$\theta^{i}(j+1, G) = \theta^{i}(j, G) + \beta - 1(\theta^{r_1}(j, G) - \theta^{r_2}(j, G) \quad (13)$$

where represents user-specified value greater than 1 for use by MBFOA's swarming operator. $\theta^{r_2}(j, G)$ and $\theta^{r_2}(j, G)$ are swarm's randomly selected bacteria and ($i \neq r_1 \neq r_2$), respectively. Twirl operators use the positions of these two bacteria to decide on search directions and start to twirl from the bacteria $\theta^{i}(j, G)$.

Figure 6 depicts the twirl operator's behaviour with two decision variables in the range [−5, 5]. On completion of the twirl the new position of the bacterium will slip into the purple spot that bact1 and bact2 specify and are referred to as $\theta^{r_2}(j,)$G, and $\theta^{r_2}(j, G)$, correspondingly.

The ideal bacterium is inserted, showing that this operator is working to get all of the search space's areas (i.e. not those present in the neighbourhood of the best current solution as it is pushed by the swarming movement).



Bact2
(B − 1) ∗ (bact1 − bact2)
Bact1
▲ Bacterium in process
◆ New bacterium after mutation
● Randomly selected bacteria
□ Best bacterium of swarm

Figure 6. A graphical example of the first exploration twirl.

(2) Exploitation twirl

The second twirl reverts to the original twirl based on random search directions, but it is now merged with small random step size values so that smooth motions may be precisely determined, as illustrated below:

$$\theta^{i}(j+1, G) = \theta^{i}(j, G) + C(i, G)\varphi(i) \quad (14)$$

where the step size values consist of an n-dimensional random vector referred to again (i, G), computed at every generation as given below:

$$C(i, G)_k = R * \Delta_{(i)_{k'}} \quad k = 1, \ldots, n \quad (15)$$

where $\Delta_{(t)}$ refers to an arbitrarily produced value exhibiting uniform distribution within [Lk, Uk] of decision variable k. R indicates a user-specified parameter for increasing the step size, and its value must be almost zero, for instance, $5.00E-03$. During the first cycle, the computation of the step size is done with only $\Delta_{(t)}$ to let the bacteria in the initial swarm traverse in various directions within the search space and escaping attractors when the process starts.

Figure 7 depicts the twirl behaviour where the green triangle representing the bacterium can move in any random search direction but near its current position irrespective of other bacterial positions in the swarm.

The combined effect of both suggested twirls with the swarming operator, with all three inside the chemotaxis process, demonstrates an increased potential to escape the local optimal solutions and encourages a rapid convergence. This result is feasible because TT-MBFOA includes a twirl for exploration (the initially suggested twirl), a twirl for choosing convergence (the MBFOA swarming operator) and a fine twirl to boost the excellent quality solutions (the second proposed twirl).

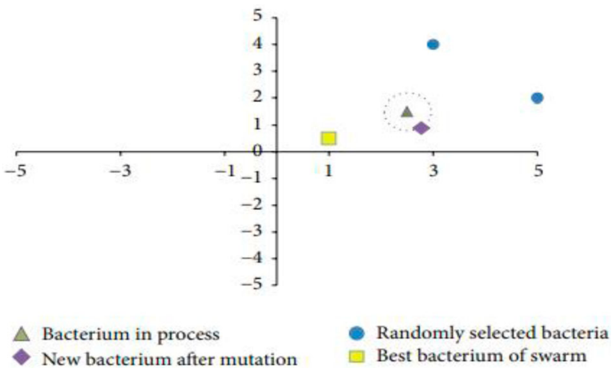


Figure 7. Random twirl behaviour.

These suggested TT-MBFOs are very resilient and efficient in obtaining the optimal coefficient of FIRs. Power usage may be lowered in this manner as well.

CSEs algorithm

Strength reduction at the algorithm level reduces the number of additions and multiplications in arithmetic circuits. One such numerical technique is called sub-expression elimination [31]. This technique improves the speed, power and area of the circuit greatly. This strength reduction reduces the total capacitance and, therefore, reduces the power consumption. The sub-expression elimination method is used over expressions which have a set of common multiplicands. It identifies recursive occurrences of identical bit patterns in the coefficients.

To facilitate comparison of the transposed with the original, the input and output signals remain "switched", so that signals generally flow right-to-left instead of the usual left-to-right. The FIRs are given by Equation (1). The variable x is calculated using just the coefficient h at different time point k. Consider the following:

$$y(n) = \sum_{k=0}^{N-1} h(k) \times (n-k) \tag{16}$$

The goal of CSEs [32] is to find the filter coefficient's common factors which are shared multiplication blocks (MBs) in transposed form, as shown in Figure 8. Pooling MBs allow for decreases in the total sizes of FIR circuits. This section elaborates on the use of novel CSEs for extracting the coefficient's common factors in CSD notations. The programme obtains data on the frequency of occurrences of coefficients and their inverses for searching common components. The pseudo-code for the proposed the CSD-based common sub-expression elimination (CCSE) method is shown in Algorithm 1.

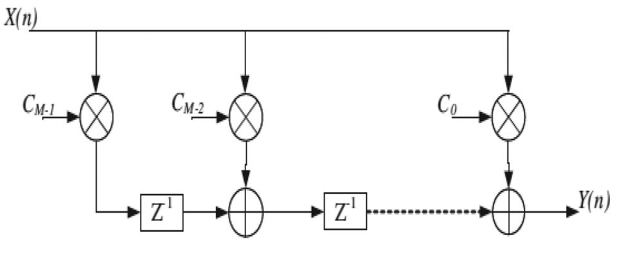


Figure 8. Transposes direct form structure.

Algorithm 1 Pseudo-code of the proposed CCSE algorithm

```
Input: Set boundary conditions 0 ≤N≤ (filter's order −1)
Output: Optimal coefficients with the same Common Sub-expression (CS)
   value
Define Filter Tap;
Max = Filter Tap-1;
//Set conditional terminations
While (true)
{
N = 0; MaxSECnt = 0; //Initialize Variable.
 While (N < = Max)
{
Record Table ← Count SE (Filter Coefficient); //Step 1
N = N+1; //Step 2
}
Record Table ← Inverse Count (Record Table);
//Step 3
For (i = 0; i < Record Table.size; i++)
MaxSECnt = Max (Record Table[i].Counts, MaxSECnt);
 //Step 4
If (MaxSECnt > 1) //Step 5
Filter Coefficient ← Simplify Coefficient (Filter Coefficient);
 Else
Break; //Step 5 terminated
 }
```

Initialization is fixing boundaries $0 \leq N \leq$ (filter's order -1) and setting $= 0$.

Step 1: Get the Nth coefficient and all of the coefficient's nonzero bit locations (from high to low). Save these places and look for Sub-expression elimination (SE) combinations with multiple non-zero bits. To simplify, all combinations are employed in the form of Basic -expression eliminations (BEs) (list the BEs). When input elements match table BE's frequencies, statistical frequencies of BEs are increased. When they do not match, they are added as a new BE to the table.

Step 2: Initializing $N = N + 1$ and determining if N is greater than the set values of boundary condition during initializations. When it is lesser, Step 1 is repeated, else Step 3 is executed.

Step 3: Evaluate all table BEs to get their corresponding and reciprocal SEs. When SEs are reversed, positive SEs are used as fundamental elements. Determining negative SEs instance counts.

Step 4: Evaluate all table BEs to assess the highest occurrences. When the maximum frequency is 1, executions proceed to the next stage and for maximum frequencies > 1, BEs are chosen as CSs and if it surpasses BEs with the same maximum frequency and frequency > 1, shorter BEs are chosen as extracted CSs.

Step 5: Collect all coefficients with the same CSs as those produced in Step 4 and those with the respective inverted CSs, and carry out the elimination procedure. When the procedure is complete, a new variable is

Table 1. Specification of the benchmark FIRs designs.

Selected FIRs	Type	Taps	$\omega_{s2}(\pi)$	$\omega_{p1}(\pi)$	$\omega_{p2}(\pi)$	$\omega_{s2}(\pi)$	NS_1	NS_2	NS_3	P_1	P_2	P_3
FIR1	LP	51			0.41	0.51	14	37	0	27.45%	72.55%	
FIR2		102			0.64	0.69	40	62	0	39.22%	60.78%	
FIR3		254			0.23	0.25	146	108	0	57.48%	42.52%	
FIR4		507			0.95	0.96	436	71	0	86.00%	14.00%	
FIR5	HP	57	0.11	0.21			18	39	0	31.58%	68.42%	
FIR6		111	0.25	0.30			48	63	0	43.24%	56.76%	
FIR7		277	0.49	0.51			154	123	0	55.60%	44.40%	
FIR8		551	0.79	0.80			418	133	0	75.86%	24.14%	
FIR9	BP	51	0.15	0.25	0.3	0.4	16	35	0	31.37%	68.63%	
FIR10		102	0.53	0.58	0.68	0.73	44	58	0	43.14%	56.86%	
FIR11		254	0.63	0.65	0.7	0.72	160	94	0	62.99%	37.01%	
FIR12		507	0.82	0.83	0.88	0.89	366	141	0	72.19%	37.81%	
FIR13	BS	57	0.16	0.26	0.31	0.41	22	34	1	38.60%	59.65%	1.75%
FIR14		111	0.35	0.40	0.50	0.55	64	46	1	57.66%	41.44%	0.9%
FIR15		277	0.56	0.58	0.68	0.70	194	83	0	70.04%	29.96%	
FIR16		551	0.82	0.83	0.88	0.89	468	83	0	84.96%	15.06%	

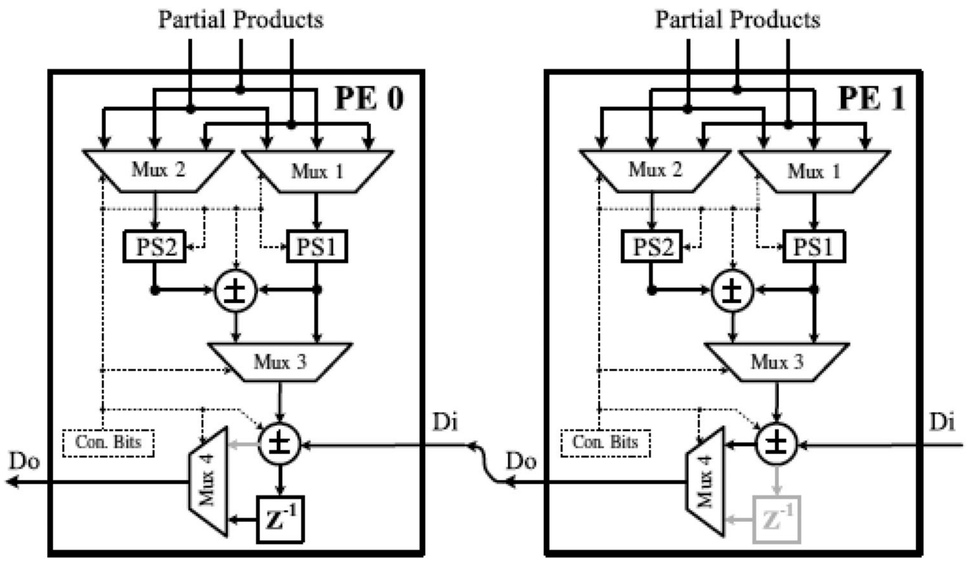


Figure 9. Operation of two cascaded PEs.

inserted back into the real expressions, the loop value N_i is set to zero and the process is returned to Step 1.

The suggested optimization technique is adaptable, resilient and easily manageable with non-differential objective functions related to RFIRs. This is the most effective co-efficient optimization and hardware complexity approach. The findings of the experiments are presented in the next section.

4. Results and discussion

This section demonstrates the proposed MBFO-CSE algorithm using an example. The RFIRs were designed with MBFO-CSE and implemented in the TDF structure for various word lengths. With the aid of the Verilog code, Modelsim software was utilized to validate the RFIR design. The benchmark filter parameters are shown in Table 1. The normalized pass band and stop band edge frequencies were 0.3 π and 0.5 π, respectively. The ripple values in the pass band and stop band were both 0.00316. The filter was designed to work with a wide range of word lengths and ordering.

Table 2. Adders and shifters counts of RFIRs.

Architecture	Distributed arithmetic	Multiple constant multiplications	MBFOA-CSE
Adders	6	6	4
Shifters	6	5	4

The aim function specified the filter criteria. Two cascaded PEs involve operations for three or four CSs, as shown in Figure 9. The two cascaded PEs may perform coefficient multiplications for three or four CSs. As shown in Figure 9, the output *Do* of PE1 is linked to the input *Di* of PE0. Every additional configuration, except the *Mux*4 in each PE, may be defined in the same way as in Case 1. The output of adder/subtractor 2 is transmitted to the Mux4 of PE1 through the *Di* of PE0, and the Mux4 of PE0 propagates the output of the register (Z^{-1}) to the next step. As a result, an integrated PE0 and PE1 operation may perform coefficient multiplication for nearly any four coefficients.

Tables 1 and 2 illustrate the design specification of the RFIRs.

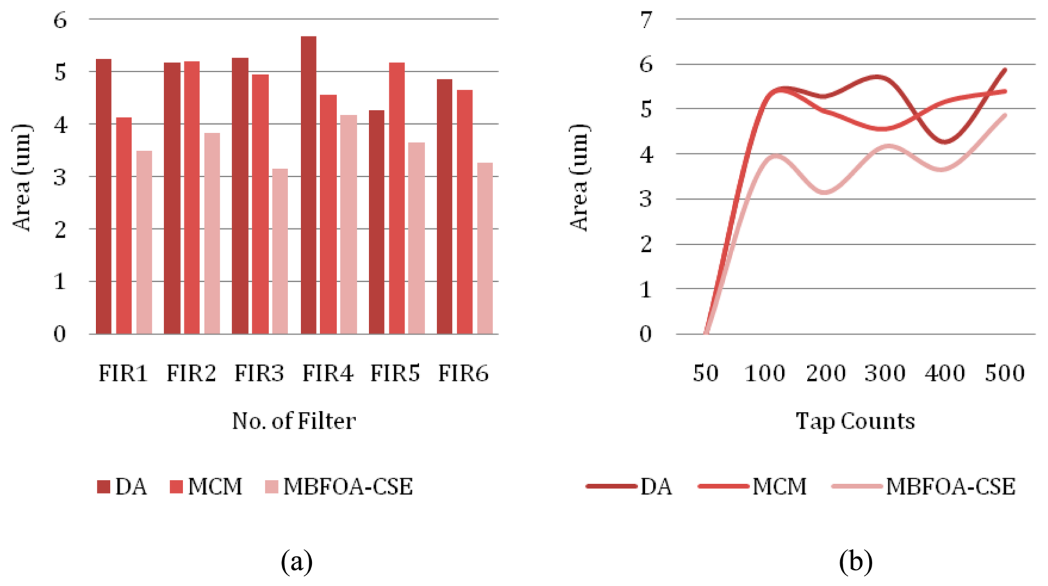


Figure 10. Area comparisons between the proposed and existing RFIR designs. (a) Number of filter vs. area and (b) area vs. varying with tap counts.

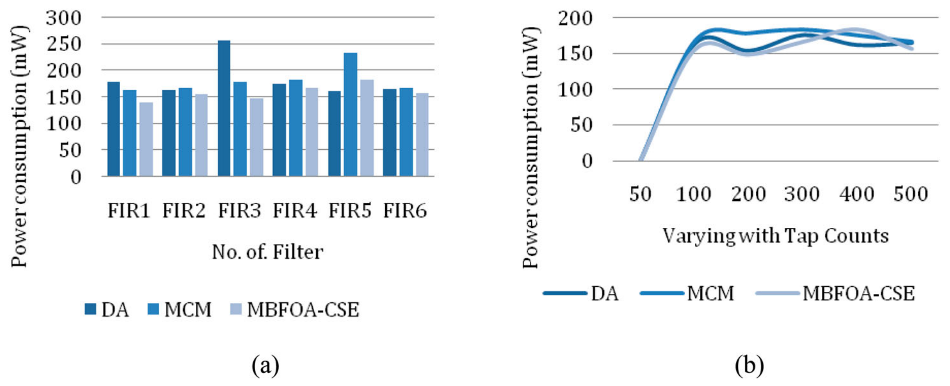


Figure 11. Power consumption comparisons between the proposed and existing RFIR designs. (a) Number of filter vs. power consumption and (b) power consumption vs. varying with tap counts.

Figure 10 illustrates the synthesized area attained for three kinds of designs, referred to as the DA-based filters, Multiple constant multiplication (MCM)-based filters and the proposed MBFOA-CSE-based RFIR designs over 16 sets of specifications. It shows that the proposed MBFOA-CSE design must have an edge over the existing design concerning area reduction, irrespective of the filter order. For instance, nearly 55.4% of hardware resources are reduced in the proposed filter design compared to the existing design, in the case of 551-tap FIRs (in 16-bit). About the DA-based filters, MCM-based filters, the proposed MBFOA-CSE must

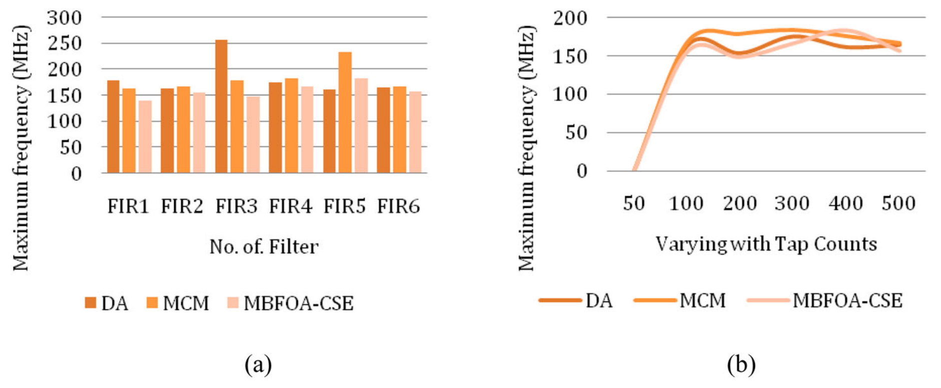


Figure 12. Maximum frequency comparisons between the proposed and existing RFIR designs. (a) Number of filter vs. maximum frequency and (b) maximum frequency vs. varying with tap counts.

have nearly identical area takeover, if the tap count is below 254. But, as the tap count increases, the savings in the area are also increased in the proposed MBFOA-CSE design.

Figure 11 illustrates the power consumption values achieved for the synthesized designs provided, known as the DA-based filters, MCM-based filters and the proposed MBFOA-CSE-based RFIR design realizations over 16 sets of filter specifications. In the case of lower-order filters, the power dissipation is matchable among the three designs. If the filter order is greater than 255, the proposed MBFOA-CSE design intends to consume much reduced power compared to DA-based filters and MCM-based filter designs.

In Figure 12, the maximum frequency for the three contemporary designs is compared. The count of the adder utilized in the proposed MBFOA-CSE design is higher than the count in the available designs. Therefore, the maximum clock frequency of the proposed MBFOA-CSE design is the least among all the three structures. Since fewer partial products are required in the proposed MBFOA-CSE design in its calculation, the maximum working frequency is greater in comparison with the other two architectures. The filter order and coefficient optimization are effectively utilized with the help of the proposed MBFOA along with the CSE-based design in this work. On average, the proposed MBFOA-CSE design achieves nearly 58.4% (the highest one) and 13.4% (the least one) rise in the maximum frequency correspondingly compared to the DA and the MCM designs.

5. Conclusion

This paper has described a design strategy for hardware-optimized RFIR filters using MBFOs and CSEs. The filter was designed using MBFOs for different filter orders, and the implementation was done in a TDF structure. The validation of the RFIR structure was carried out in Modelsim software using the Verilog code. The enhanced RFIRs coefficient optimization approach has been presented, which adds some extra non-zero digits to the CSD codes to balance the non-uniform distribution while reducing the hardware complexity required to create equal rippled FIRs. This technique begins with creating the filter using the usual equal rippled method to attain a minimal order. The order is optimized until the criteria are precisely met, at which point the filter's grouping of CSD coefficients are quantized. The MBFOs find the initial set of filter coefficients using a restricted amount of SPT terms while retaining filter response quality. On obtaining coefficients, CSEs are used and hardware complexity is determined in terms of adder counts. Finally, the suggested reconfigurable filter design was compared to the best works already available in terms of area, power consumption and frequency. According to the preceding research, when the filter order exceeds 255 taps, the suggested MBFOA-CSE design expects to utilize significantly less power than DA-based filters and MCM-based filter designs. Also it shows that nearly 55.4% of hardware resources are reduced in the proposed filter design compared to the existing design, in the case of 551-tap FIRs (in 16-bit). Hence, the proposed MBFOA-CSE-based system outperforms previous reconfigurable FIR solutions in terms of performance and hardware complexity. The suggested approach focuses largely on enhancing filter performance. Future research might investigate its dynamically reconfigurable techniques.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

[1] Pathan A, Memon TD, Keerio S, et al. FPGA based performance analysis of multiplier policies for FIR filter. IEEE International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES); 2016. p. 17–20.

[2] Paul A, Khan TZ, Podder P, et al. Reconfigurable architecture design of FIR and IIR in FPGA. IEEE International Conference on Signal Processing and Integrated Networks (SPIN); 2015. p. 958–963.

[3] Kumar DA, SaiKumar M, Samundiswary P. Design and study of a modified parallel FIR filter using fast FIR algorithm and symmetric convolution. IEEE International Conference on Information Communication and Embedded Systems (ICICES2014); 2014. p. 1–6.

[4] Abbaszadeh A, Azerbaijan A, Sadeghipour KD. New hardware efficient reconfigurable FIR filter architecture suitable for FPGA applications. IEEE International Conference on Digital Signal Processing (DSP); 2011. p. 1–4.

[5] Ali N, Garg B. New energy efficient reconfigurable FIR filter architecture and its VLSI implementation. International Symposium on VLSI Design and Test; 2017. Vol. 711, p. 519–532. doi:10.1007/978-981-10-7470-7_51

[6] Rao BRS, Sundari BBT. An efficient reconfigurable FIR filter for dynamic filter order variation. IEEE International Conference on Communication and Electronics Systems (ICCES); 2019. p. 1724–1728.

[7] Kumm M, Möller K, Zipf P. Dynamically reconfigurable FIR filter architectures with fast reconfiguration. IEEE International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC); 2013. p. 1–8.

[8] Lee SJ, Choi JW, Kim SW, et al. A reconfigurable FIR filter architecture to trade off filter performance for dynamic power consumption. IEEE Trans Very Large Scale Integration (VLSI) Systems. 2010;19(12): 2221–2228.

[9] Ozalevli E, Huang W, Hasler PE, et al. A reconfigurable mixed-signal VLSI implementation of distributed arithmetic used for finite-impulse response filtering. IEEE Trans Circuits Syst Regul Pap. 2008;55(2):510–521. doi:10.1109/TCSI.2007.913735

[10] Mirzaei S, Kastner R, Hosangadi A. Layout aware optimization of high-speed fixed coefficient FIR filters

for FPGAs. Int J Reconfigur Comput. 2010;2010:1–17. doi:10.1155/2010/697625

[11] Agrawal N, Kumar A, Bajaj V, et al. Design of digital IIR filter: a research survey. Appl Acoust. 2021;172:107669. doi:10.1016/j.apacoust.2020.107669

[12] Ma RX, Wang YH, Hu W, et al. Optimum design of multistage half-band FIR filter for audio conversion using a simulated annealing algorithm. 13th IEEE Conference on Industrial Electronics and Applications; 2018. p. 74–78.

[13] Miyata T, Asou H, Aikawa N. Design method for FIR filter with variable multiple elements of stopband using genetic algorithm. 23rd IEEE International Conference on Digital Signal Processing; 2018; p. 8631793.

[14] Dash J, Dam B, Swain R. Design of multipurpose digital FIR double-band filter using hybrid firefly differential evolution algorithm. Appl Soft Comput. 2017;59:529–545. doi:10.1016/j.asoc.2017.06.025

[15] Tsutsumi S, Suyama K. Design of FIR filters with discrete coefficients using ant colony optimization. Electron Commun Jpn. 2014;97(4):30–37. doi:10.1002/ecj.11522

[16] Sarangi SK, Panda R, Das PK, et al. Design of optimal high pass and band stop FIR filters using adaptive cuckoo search algorithm. Eng Appl Artif Intell. 2018;70:67–80. doi:10.1016/j.engappai.2018.01.005

[17] Mohanty BK, Meher PK. High-performance energy-efficient architecture for FIR adaptive filter based on the new distributed arithmetic formulation of block LMS algorithm. IEEE Trans Signal Process. 2013; 61(4):921–932. doi:10.1109/TSP.2012.2226453

[18] Guo R, DeBrunner LS. Two high-performance adaptive filter implementation schemes using distributed arithmetic. IEEE Trans Circuits Syst Express Briefs. 2011;58(9):600–604.

[19] Gunasekaran K, Manikandan M. Low power and area efficient reconfigurable FIR filter implementation in FPGA. IEEE International Conference on Current Trends in Engineering and Technology (ICCTET); 2013: p. 300–303.

[20] Haridas G, George DS. Area efficient low power modified booth multiplier for FIR filter. Procedia Technol. 2016;24:1163–1169. doi:10.1016/j.protcy.2016.05.070

[21] Ramachary K, Siva LR. Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic. Int J Scientif Eng Technol Res. 2015;4(25): 4731–4737.

[22] Nishok VS, Poongodi P, Vijeyakumar KN. Design and performance estimation of efficient approximate carry select adder. Appl Math Inf Sci Int. J. 2018;12(6): 1219–1225.

[23] Vignesh O, Mangalam H. Low power binomial coefficient architecture for unused spectrum detector. Analog Integr Circuits Signal Process. 2019;99(3):599–606.

[24] Kadam M, Sawarkar K, Mande S. Investigation of suitable DSP architecture for efficient FPGA implementation of FIR filter. IEEE International Conference on Communication, Information & Computing Technology (ICCICT); 2015. p. 1–4.

[25] Pristach M, Dvorak V, Fujcik L. Enhanced architecture of FIR filters using block memories. IFAC-Papers OnLine. 2015;48(4):306–311. doi:10.1016/j.ifacol.2015.07.052

[26] Jia R, Yang HG, Lin CY, et al. A computationally efficient reconfigurable FIR filter architecture based on coefficient occurrence probability. IEEE Trans Comput Aided Des Integr Circuits Syst. 2015;35(8):1297–1308. doi:10.1109/TCAD.2015.2504922

[27] Sriadibhatla S, Baboji K. Design and implementation of area and power efficient reconfigurable fir filter with low complexity coefficients. Gazi Univ J Sci. 2019;32(2):494–507.

[28] Lian L, Tian Z. FIR digital filter design based on improved artificial bee colony algorithm. Soft Comput. 2022;26(24):13489–13507. doi:10.1007/s00500-022-07506-w

[29] Arumugam N, Paramasivan B. An integrated FIR adaptive filter design by hybridizing canonical signed digit (CSD) and approximate booth recode (ABR) algorithm in DA architecture for the reduction of noise in the sensor nodes. Multidimens Syst Signal Process. 2021;32(4):1277–1311. doi:10.1007/s11045-021-00783-y

[30] Passino KM. Bacterial foraging optimization. Int J Swarm Intellig Res (IJSIR). 2010;1(1):1–16. doi:10.4018/jsir.2010010101

[31] Mahesh R, Vinod AP. A new common sub expression elimination algorithm for realizing low-complexity higher order digital filters. IEEE Trans Comput Aided Des. 2008;27(2):217–229. doi:10.1109/TCAD.2007.907064

[32] Liacha A, Oudjida AK, Bakiri M, et al. Radix-2 r recording with common subexpression elimination for multiple constant multiplications. IET Circuits Devices Syst. 2020;14(7):990–994. doi:10.1049/iet-cds.2020.0213