mathematics

MDPI

Article

A Compact and High-Performance Acoustic Echo Canceller Neural Processor Using Grey Wolf Optimizer along with Least Mean Square Algorithms

Eduardo Pichardo *, Esteban Anides *, Angel Vazquez, Luis Garcia, Juan G. Avalos, Giovanny Sánchez, Héctor M. Pérez and Juan C. Sánchez

Instituto Politécnico Nacional, ESIME Culhuacan, Av. Santa Ana No. 1000, Ciudad de México 04260, Mexico
* Correspondence: edua_95pim@hotmail.es (E.P.); eanidesc1800@alumno.ipn.mx (E.A.); Tel.: +52-55-2101-9551 (E.P. & E.A.)

Abstract: Recently, the use of acoustic echo canceller (AEC) systems in portable devices has significantly increased. Therefore, the need for superior audio quality in resource-constrained devices opens new horizons in the creation of high-convergence speed adaptive algorithms and optimal digital designs. Nowadays, AEC systems mainly use the least mean square (LMS) algorithm, since its implementation in digital hardware architectures demands low area consumption. However, its performance in acoustic echo cancellation is limited. In addition, this algorithm presents local convergence optimization problems. Recently, new approaches, based on stochastic optimization algorithms, have emerged to increase the probability of encountering the global minimum. However, the simulation of these algorithms requires high-performance computational systems. As a consequence, these algorithms have only been conceived as theoretical approaches. Therefore, the creation of a low-complexity algorithm potentially allows the development of compact AEC hardware architectures. In this paper, we propose a new convex combination, based on grey wolf optimization and LMS algorithms, to save area and achieve high convergence speed by exploiting to the maximum the best features of each algorithm. In addition, the proposed convex combination algorithm shows superior tracking capabilities when compared with existing approaches. Furthermore, we present a new neuromorphic hardware architecture to simulate the proposed convex combination. Specifically, we present a customized time-multiplexing control scheme to dynamically vary the number of search agents. To demonstrate the high computational capabilities of this architecture, we performed exhaustive testing. In this way, we proved that it can be used in real-world acoustic echo cancellation scenarios.

Keywords: grey wolf optimization; swarm intelligence; real world application; spiking neural P system; AEC system; LMS; neuromorphic architecture; FPGA

MSC: 68W10; 68Q06; 68Q45; 68W99; 94A12

Citation: Pichardo, E.; Anides, E.; Vazquez, A.; Garcia, L.; Avalos, J.G.; Sánchez, G.; Pérez, H.M.; Sánchez, J.C. A Compact and High-Performance Acoustic Echo Canceller Neural Processor Using Grey Wolf Optimizer along with Least Mean Square Algorithms. *Mathematics* **2023**, *11*, 1421. https://doi.org/10.3390/math11061421

Academic Editor: Gaige Wang

Received: 1 February 2023
Revised: 28 February 2023
Accepted: 13 March 2023
Published: 15 March 2023

1. Introduction

Nowadays, acoustic echo canceller (AEC) systems mainly use the least mean square (LMS) adaptive filter algorithm, since it exhibits low computational complexity [1]. Therefore, this algorithm is easy to implement in low-area devices. However, its use can cause instability in the system, since it has an uni-model error surface. Hence, this algorithm is limited when a multimodal error surface is considered, since this algorithm must be initialized in the valley of the global optimum to converge to the global optimum [2]. Another aspect is linked to its convergence speed, because this depends on the eigen-value spread of the correlation Matrix (R) [3]. To overcome these problems, bio-inspired evolutionary [4,5] and swarm intelligence (SI) techniques [6–8], have emerged as potential solutions for the parameter optimization. Recently, the grey wolf optimization (GWO)

algorithm [6], which is considered to be one of the most recent metaheuristic SI methods, has proven to be very successful in multiple fields, such as image processing [9,10], health care/bioinformatics [11,12], control systems [13], electromagnetics [14,15], environmental applications [16,17], and adaptive filtering [18,19], among others [20,21], since this algorithm offers impressive characteristics in contrast with other swarm intelligence methods.

In general terms, the GWO algorithm shows a good balance between the exploration and exploitation processes during the search. As a consequence, this allows high accuracy. From the engineering point of view, this algorithm can be seen as a potential solution in practical applications, since it involves fewer parameters and less memory compared with the most advanced metaheuristic SI methods. Therefore, this algorithm requires few controlling parameters. As a consequence, its practicality is increased significantly [21]. Based on these features, new variants of the GWO algorithm can be developed to be applied in practical and real-time acoustic echo canceller (AEC) systems.

In general, bio-inspired algorithms are capable of globally optimizing any class of adaptive filter structures. Therefore, the use of these algorithms opens new opportunities in the development of advanced adaptive filters and enables their implementation in embedded devices [22]. Specifically, recent studies have proven that the particle swarm optimization (PSO) algorithm can be used in the development of AEC systems. For example [23] proposed a PSO-based AEC system to guarantee a high degree of accuracy in terms of echo return loss enhancement (ERLE). Specifically, the authors used the PSO to perform the error minimization in the frequency domain. Ref. [24] presented a PSO-based AEC system, in which the PSO algorithm provides a very fast convergence rate by performing the error minimization in the time domain. Another PSO-based AEC system was developed by [25]. This approach was applied to multichannel systems to improve stability.

After analysing all the works mentioned above, we found that the PSO algorithm may suffer some limitations, especially when a heavy constraint optimization is required. Under this situation, one may get trapped in local minima. To avoid this, the GWO algorithm can be seen as a potential solution, since it offers a high convergence rate at the cost of exhibiting high computational complexity and low tracking capabilities [26–28]. In particular, good performance can be obtained, especially when a large population is used, i.e., population-based meta-heuristics generally have greater exploration when compared to a single solution-based algorithm [6]. However, its implementation in current embedded devices becomes infeasible. Hence, the development of new variants of GWO algorithms, to decrease their computational cost whilst maintaining performance, is still a great challenge. Recently, some authors have proposed convex combinations of adaptive filters, in which two adaptive algorithms with complementary capabilities are used to improve the steady-state MSE and the tracking performance [29,30]. With respect to the latter, the tracking performance determines the capabilities of the system to track variations in the statistics of the signals of interest [31]. Therefore, in practical AEC systems, the improvement of this factor is highly required [32].

Here, we propose a new variant of the convex adaptive filter, based on the GWO and LMS algorithms. In addition, we include the block-processing scheme in both algorithms to easily implement them in parallel hardware architectures. As a consequence, these algorithms were applied to real-time and practical AEC applications. Specifically, we used the GWO algorithm to guarantee a high convergence rate and high tracking capabilities. With respect to the latter, we added new exploration capabilities by dynamically adjusting the search space, especially in the context of abrupt changes occurring in the acoustic environment, which cannot be achieved when using the conventional GWO algorithm. A significant improvement in terms of computational cost was achieved by dynamically decreasing the population of the GWO algorithm over the filtering process. In addition, the use of the LMS allowed us to improve the steady-state MSE.

From an engineering perspective, the development of low computational complexity metaheuristic SI methods makes their implementation in current resource-constrained devices feasible. In addition, the development of advanced and novel implementation

techniques also opens new opportunities in the creation of compact and high-performance devices. Specifically, the simulation of the proposed convex GWO/LMS adaptive filter requires an enormous number of large precision multipliers. Therefore, the development of low area, low latency and large precision adders and multipliers is still a challenging task. Inspired by the neural phenomena, Ionescu presented, for the first time, a class of distributing and parallel computing models, denominated as spiking neural P systems [33]. This new area of membrane computing intends to exploit, to the maximum, the intrinsic parallel capabilities of the soma of the neurons to create novel processing systems. In recent years, several authors focused their efforts to create advanced arithmetic circuits, such as adders and multipliers.

- *Parallel neural adder*.
  Recently, Ref. [34] developed a neural adder circuit to compute two large integer numbers in parallel. In spite of this achievement, this adder demands an enormous number of synapses and neurons to process large integer numbers. As a result, its use for simulation purposes becomes impractical, since metaheuristic SI methods demand high precision numerical accuracy.
- *Parallel neural multiplier*.
  In [35], the authors intended to significantly reduce the number of synapses to create an ultra-compact parallel multiplier. Despite decreasing the area consumption, the processing speed was still high.

Analyzing previous works, we noted that these circuits were proposed to process large integer numbers in parallel at the cost of increasing their processing speed. Therefore, any of these circuits can be used in the simulation of real-time AEC systems. In addition, arithmetic circuits with more precision exhibit a clear trade-off between area consumption and processing speed. From an engineering perspective, several authors still continue to make tremendous efforts to design advanced neural arithmetic circuits with high-precision to be used in real-time AEC systems. Specifically, these developments face two large challenges:

- Design of high-speed and high-precision neural adders and neural multipliers.
- Design of high-processing speed hardware architectures to efficiently simulate the proposed convex GWO/LMS adaptive filter in embedded devices.

Regarding the latter, we developed three potential proposals:

1. *Design of a high-precision floating-point parallel adder circuit*. Here, we present, for the first time, a neural adder, which computes the numbers in a customized floating-point model. We employ new variants of the SN P systems, called coloured-spikes [36], rules on the synapses [37], target indications, extended channel rules [38] and extended rules [39] to process numbers under this format in the proposed arithmetic circuit.
2. *Design of a high-precision floating-point parallel multiplier*. Here, we present, for the first time, the development of a neural multiplier to compute floating-point numbers at high processing speeds.
3. *Design of a new FPGA-based GWO/LMS neuromorphic architecture*. We design the proposed neuromorphic architecture employing basic digital components, such as shift registers, adders and multiplexors, to guarantee a low area consumption. Since the proposed convex GWO/LMS adaptive filter dynamically varies the number of search agents, we propose, for the first time, a time-multiplexing control scheme to adequately support this behavior.

Our results showed that the proposal of new variants of the GWO algorithm, along with new implementation techniques, generates a compact neuromorphic architecture to be used in practical and real-time AEC applications.

The paper is structured as follows. Section 2 introduces the fundamentals of GWO and LMS algorithms. Additionally, we introduce the proposed Convex GWO/LMS algorithm, which involves the use of a new set of equations to improve its tracking capabilities and computational complexity. Section 3 presents software simulation tests, including the justification of the selection of some tuning parameters and display comparisons

between existing approaches and the proposed algorithm. In Section 4, we present a new parallel neural adder circuit and a new parallel neural multiplier circuit, which are highly demanded in the computation of the proposed algorithm. In addition, this section introduces experimental results of the implementation of the proposed Convex GWO/LMS algorithm in the Stratix IV GX EP4SGX530 FPGA. Finally, in Section 5 we provide the conclusions.

2. The Proposed Block Convex GWO/LMS Algorithm

2.1. GWO Algorithm

In general terms, the GWO is considered a population-based optimization technique inspired by the behavior of the Canis lupus [6]. This algorithm intends to mimic the hunting and hierarchical behavior of grey wolves. Regarding the latter, this algorithm involves four hierarchical levels; alpha (α) represents the fittest solution in the population, while beta (β) and delta (δ) denote the second and third best solutions, respectively. Finally, omega (ω) are the search agents. To simulate the GWO algorithm, the following equations are used:

$$\overrightarrow{W}(n+1) = \overrightarrow{W}_p(n) - \overrightarrow{A} \cdot |\overrightarrow{C} \times \overrightarrow{W}_p(n) - \overrightarrow{W}(n)| \tag{1}$$

where n denotes the current iteration, $\overrightarrow{W}_p$ is the position vector of the prey, $\overrightarrow{W}$ represents the position of a grey wolf, $\overrightarrow{A} = 2\overrightarrow{a} \cdot \overrightarrow{r}_1 - \overrightarrow{a}$ and $\overrightarrow{C} = 2\overrightarrow{r}_2$ denote coefficient vectors, where $\overrightarrow{r}_1$ and $\overrightarrow{r}_2$ represent random vectors in $[0, 1]$, respectively, and $\overrightarrow{a}$ is linearly decreased from 2 to 0 using the following equation

$$\overrightarrow{a}(t) = 2 - \frac{2t}{MaxIter} \tag{2}$$

where $MaxIter$ is the total number of iterations.

To update the position of the search agents, the following equations must be used

$$\overrightarrow{W}_1(n) = \overrightarrow{W}_\alpha(n) - \overrightarrow{A}_1 \cdot |\overrightarrow{C}_1 \cdot \overrightarrow{W}_\alpha(n) - \overrightarrow{W}(n)| \tag{3}$$

$$\overrightarrow{W}_2(n) = \overrightarrow{W}_\beta(n) - \overrightarrow{A}_2 \cdot |\overrightarrow{C}_2 \cdot \overrightarrow{W}_\beta(n) - \overrightarrow{W}(n)| \tag{4}$$

$$\overrightarrow{W}_3(n) = \overrightarrow{W}_\delta(n) - \overrightarrow{A}_3 \cdot |\overrightarrow{C}_3 \cdot \overrightarrow{W}_\delta(n) - \overrightarrow{W}(n)| \tag{5}$$

$$\overrightarrow{W}_p(n+1) = \frac{\overrightarrow{W}_1(n) + \overrightarrow{W}_2(n) + \overrightarrow{W}_3(n)}{3} \tag{6}$$

where $\overrightarrow{W}_\alpha(n)$, $\overrightarrow{W}_\beta(n)$, $\overrightarrow{W}_\delta(n)$ are the best three wolves at each iteration and $\overrightarrow{W}_p(n+1)$ is the new position of the prey.

2.2. LMS Algorithm

The LMS algorithm is a practical method to obtain estimates of a filter's weights, $\mathbf{w}(n)$, in real time [1]. The equation for updating the weights of the LMS algorithm in a sample by sample fashion is given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \tag{7}$$

where $\mathbf{x}(n)$ is the current input vector, μ is a fixed step-size [0, 1], $e(n)$ is the error signal and is calculated using:

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n) \tag{8}$$

and $d(n)$ is the desired signal.

2.3. Convex GWO/LMS

As discussed above, the GWO algorithm is widely used to solve a variety of problems since it exhibits significant properties. However, it has a number of limitations and suffers

from inevitable drawbacks. The main limitation comes from the no-free-lunch (NFL) theorem, which states that no optimization algorithm is able to solve all optimization problems [40]. This means that GWO might require modification when solving some real-world problems. In particular, we proposed some modifications to the conventional GWO to be used in the development of AEC systems. Specifically, we present a new combination between the GWO and the LMS algorithms to improve tracking-capabilities and the steady-state error of the filter, respectively, since it has been proven that the use of convex combination approaches significantly improve the performance of adaptive schemes by using two adaptive algorithms [41].

As can be observed in Figure 1, the proposed system computes the coefficients of the filters, $\mathbf{w}_1$ and $\mathbf{w}_2$, as follows:

- *The calculation of the filter coefficients by employing the LMS algorithm*. Here, we use the block LMS algorithm to calculate the $\mathbf{w}_1$ weights. This has allowed us to create efficient implementations by using commercially available embedded devices [42,43]. The adaptive filter coefficients $\mathbf{w}_1$ are defined as:

$$\mathbf{w}_1(n+1) = \mathbf{w}_1(n) + \mu \mathbf{X}(n)\mathbf{e}_{LMS}(n) \tag{9}$$

where $\mathbf{w}_1(n)$ is the weight-vector $\mathbf{w}_1(n) = \begin{bmatrix} w(0), & w(1), & \cdots & w(N-1) \end{bmatrix}^T$, μ is the step-size, and $\mathbf{e}_{LMS}(n)$ is the error vector. The error vector is composed as:

$$\mathbf{e}_{LMS}(n) = \begin{bmatrix} e(nL), & e(nL-1), & \cdots & e(L(n+1)+1) \end{bmatrix}^T \tag{10}$$

and $\mathbf{X}(n)$ is derived from the current input block

$$\mathbf{x}(n) = \begin{bmatrix} x(nL), & x(nL-1), & \cdots & x(nL-N+1) \end{bmatrix} \tag{11}$$

where L depicts the length of the block and N is the size of the filter. Additionally, $\mathbf{X}(n)$ is calculated as follows:

$$\mathbf{X}(n) = \begin{bmatrix} x(nL), & x(nL-1), & \cdots & x(nL-N+1) \\ x(nL-1) & x(nL-2) & \cdots & x(nL-N) \\ \vdots & \vdots & \ddots & \vdots \\ x(nL-L+1) & x(nL-L) & \cdots & x(nL-L-N+2) \end{bmatrix}^T \tag{12}$$

The error vector is obtained as follows:

$$\mathbf{e}_{LMS}(n) = \mathbf{d}(n) - \mathbf{y}_{LMS}(n) \tag{13}$$

where the desired response vector $\mathbf{d}(n)$ is given by L

$$\mathbf{d}(n) = \begin{bmatrix} d(nL), & d(nL-1), & \cdots & d(nL-N+1) \end{bmatrix}^T \tag{14}$$

The filter output $\mathbf{y}_{LMS}(n)$ of each block is given by the following matrix vector product:

$$\mathbf{y}_{LMS}(n) = \mathbf{X}(n) \cdot \mathbf{w}_1(n) \tag{15}$$

- *The calculation of the filter coefficients by employing the GWO algorithm*. Here, the use of the block-processing scheme in SI algorithms allowed us to process the signal in real-time applications. As a consequence, this potentially allows full exploitation of the performance capabilities of the parallel hardware architectures by simulating the intrinsic parallel computational capabilities of the SI algorithms. In this work, we introduce, for the first time, the block-based GWO algorithm to be applied in practical and real-time AEC applications, as shown in Figure 2.

The encircling behavior of the grey wolves is mathematically described as follows:

$$\overrightarrow{W}(n+1) = \overrightarrow{W}_p(n) - \overrightarrow{A} \cdot |\overrightarrow{C} \cdot \overrightarrow{W}_p(n) - \overrightarrow{W}(n)| \tag{16}$$

where $\overrightarrow{A} = 2\phi(n) \cdot \overrightarrow{r}_1 - \phi(n)$ denotes a coefficient vector, and $\phi(n)$ is in function of the value of instantaneous error. In addition, this value is in a range of 0 and 2. This can be described by:

$$\phi(n) = \frac{4}{1 + e^{-[e_{GWO}(n)]}} - 2 \tag{17}$$

To obtain the best solution, a fitness function, which is defined in terms of the mean square error (MSE), is used to evaluate each search agent. Hence, the fitness value f_k of the position $\overrightarrow{W}$ is expressed as:

$$f_k(n) = \frac{1}{L} \sum_{i=1}^{L} e_k^2(i) \tag{18}$$

where $k = 1, 2, \ldots, P(n)$. Here, we propose a mechanism to dynamically adjust the number of search agents over the filtering process, i.e., $P(n)$ denotes the current number of search agents. Specifically, this adjustment is a function of the power of the instantaneous error, and is obtained as follows:

$$P(n) = \lfloor \frac{2 \cdot (P_{max} - P_{min})}{(1 + e^{-[e_{GWO}(n)])}} - (P_{max} - P_{min}) \rfloor + P_{min} \tag{19}$$

where P_{max} and P_{min} defines the maximum and the minimum number of search agents, respectively. In addition, we use Equations (3)–(6) to update the position of the search agents and the prey. On the other hand, the error signal, $e_{GWO}(n)$, and filter output, $y_{GWO}(n)$, are described as follows:

$$\mathbf{e}_{GWO}(n) = \mathbf{d}(n) - \mathbf{y}_{GWO}(n) \tag{20}$$

$$\mathbf{y}_{GWO}(n) = \mathbf{X}(n) \cdot \mathbf{w}_2(n) \tag{21}$$

where $\mathbf{w}_2(n) = \overrightarrow{W}_\alpha$.
Considering the output of both filters, $\mathbf{y}_{GWO}(n)$ and $\mathbf{y}_{LMS}(n)$ at time n, we obtain the output of the parallel filter as:

$$\mathbf{y}(n) = \lambda(n) \cdot \mathbf{y}_{GWO}(n) - [1 - \lambda(n)] \cdot \mathbf{y}_{LMS}(n) \tag{22}$$

where $\lambda(n)$ is a mixing parameter, which is in the range [0, 1]. This parameter is used to control the combination of the two filters at each iteration, and is defined by:

$$\lambda(n) = \frac{1}{e^{-a(n)}} \tag{23}$$

where $a(n)$ is an auxiliary parameter used to minimize the instantaneous square error of the filters, and is obtained as follows:

$$a(n+1) = a(n) + \mu_a \cdot e(1) \cdot \{e_{GWO}(1) - e_{LMS}(1)\} \cdot \lambda(n) \cdot [1 - \lambda(n)] \tag{24}$$

Finally, the performance of the combined filter can be further improved by transferring a portion of $\mathbf{w}_1$ to $\overrightarrow{W}_\alpha$, $\overrightarrow{W}_\beta$ and $\overrightarrow{W}_\delta$. This can be formulated as follows:

$$\overrightarrow{W}_\alpha(n) = \lambda(n) \cdot \overrightarrow{W}_\alpha(n) - [1 - \lambda(n)] \cdot \mathbf{w}_1(n) \tag{25}$$

$$\overrightarrow{W}_\beta(n) = \lambda(n) \cdot \overrightarrow{W}_\beta(n) - [1 - \lambda(n)] \cdot \mathbf{w}_1(n) \tag{26}$$

$$\overrightarrow{W}_{\delta}(n) = \lambda(n) \cdot \overrightarrow{W}_{\delta}(n) - [1 - \lambda(n)] \cdot \mathbf{w}_1(n) \tag{27}$$

In this way, the GWO filter can reach a lower steady-state MSE and continues to keep a high convergence rate.

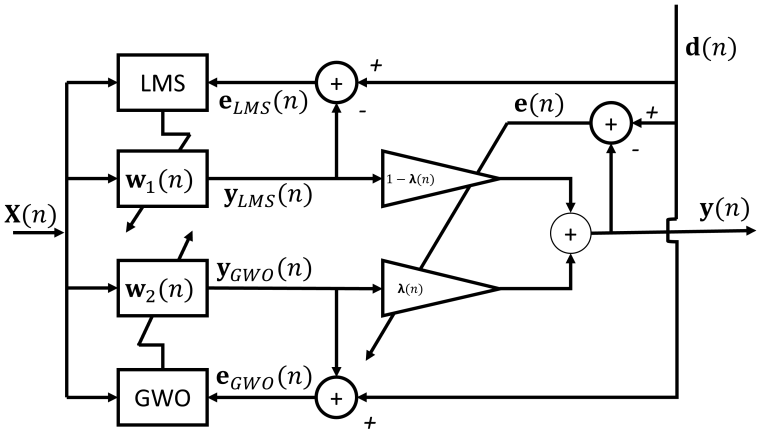


Figure 1. Proposed convex structure.

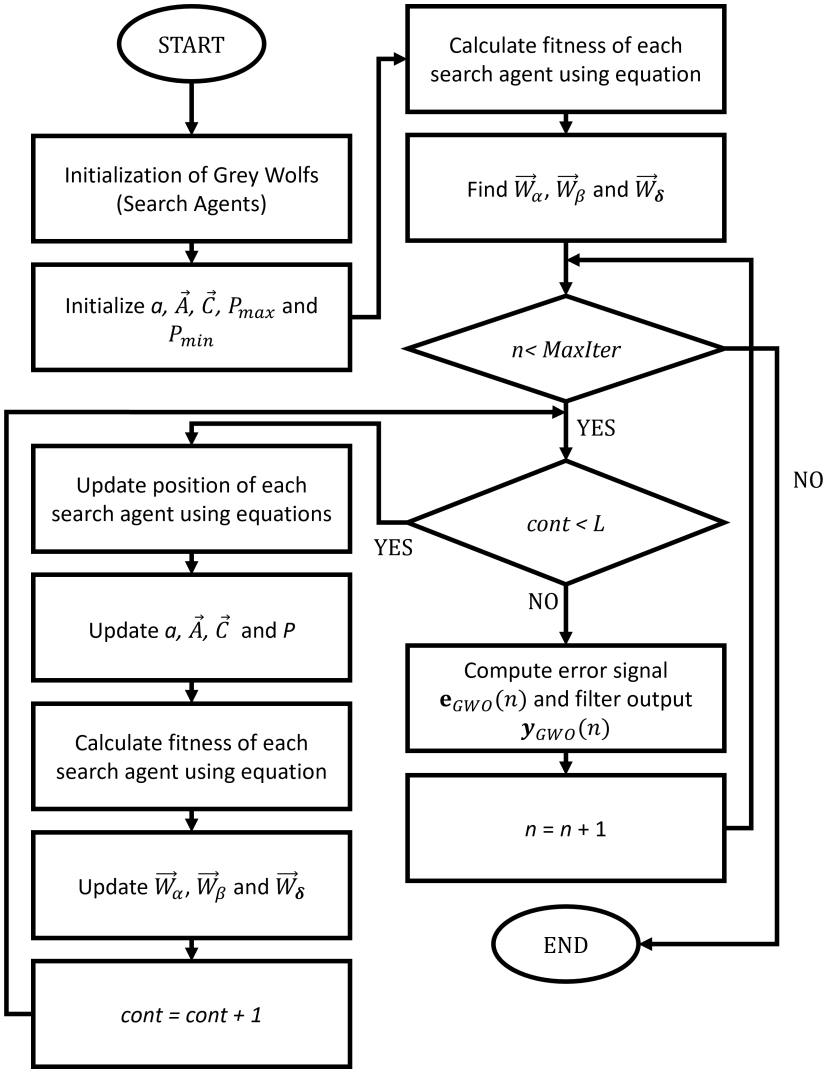


Figure 2. The flowchart of the block GWO algorithm.

3. Pure Software Simulation

Before implementing the proposed convex GWO/LMS adaptive filter in parallel hardware architectures, we simulated it in Matlab software for testing and comparison purposes. Specifically, we simulated the conventional LMS, GWO and our proposal to compare their performances. In addition, we used AEC structure, in which the existing approaches and the proposed convex GWO/LMS adaptive filter were used, as shown in Figure 3. As can be observed, $x(n)$ is the far-end input signal, $e(n)$ denotes the residual echo signal, $d(n)$ represents the sum of the echo signal, $y(n)$ and the background noise, $e_0(n)$.

To simulate the proposed convex GWO/LMS adaptive filter and existing approaches, we considered the following conditions:

1. We used an impulse response as the echo path, obtained from the ITU-T G168 recommendation [44]. This echo path was modeled using 500 coefficients, as shown in Figure 4.
2. The echo signal was mixed with white Gaussian noise (SNR = 20 dB).
3. We used an AR(1) process as input signal.
4. The filter and the block had the same length as the echo path. As is well known, the efficiency of the block processing scheme is guaranteed when the length of the blocks is greater than, or equal to, the order of the filter [45,46].
5. In the proposed algorithm, the swarm size was defined in the range of 15–30 search agents.
6. To test the tracking capabilities of the proposed algorithm, we induced an abrupt change in the impulse response of the acoustic echo path in the middle of the adaptive filtering process by multiplying the acoustic paths by -1.
7. The maximum number of iterations was set to 2,000,000.

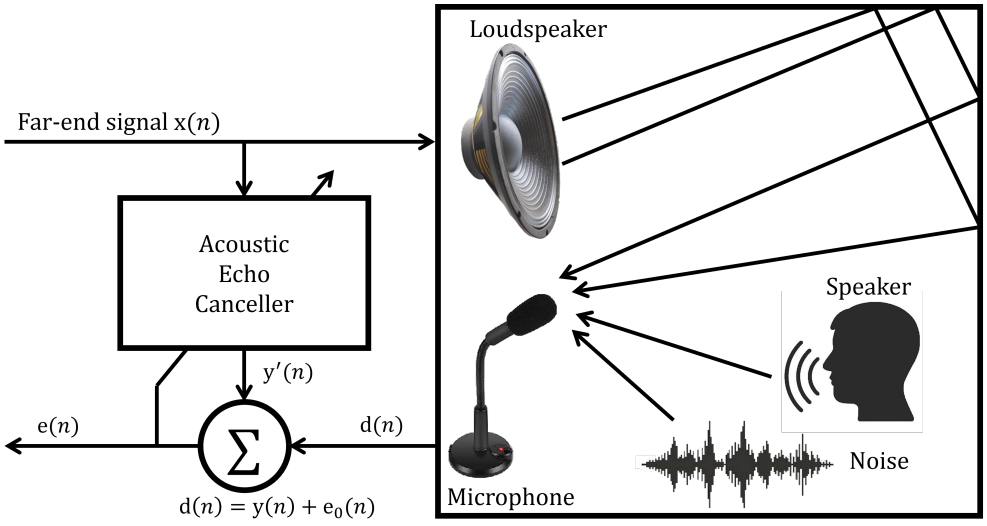


Figure 3. Structure of the acoustic echo canceller.

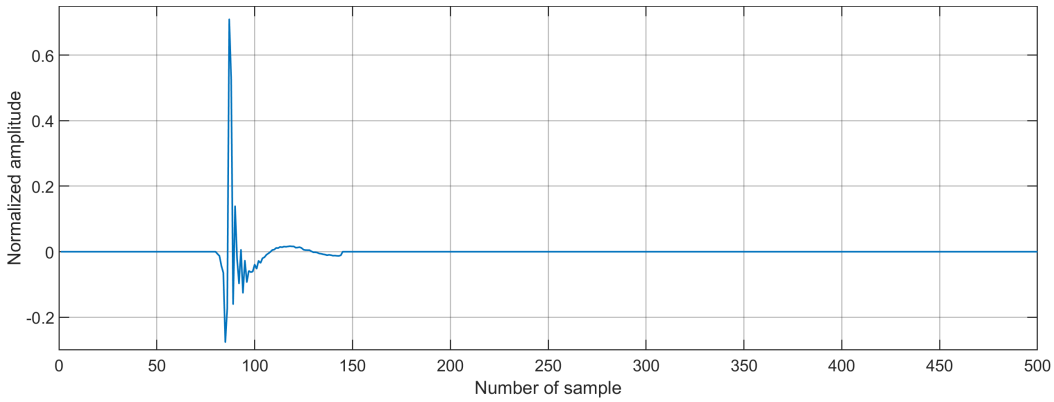


Figure 4. Acoustic echo path used for the simulation of the existing and the proposed algorithms.

Considering a single-talk scenario, we performed three experiments to verify the performance of the proposed convex GWO/LMS adaptive filter in terms of echo return loss enhancement, ($ERLE = 10log_{10}(\frac{d(n)^2}{e(n)^2})$).

- *Effect of changing the order of the adaptive filter*.
  Figure 5 shows the evaluation of the ERLE level of the proposed algorithm. In this evaluation, we used a population size of 30 search agents and varied the number of coefficients of the adaptive filter from 150 to 500. The aim of this experiment was to observe how the ERLE level was affected by using different numbers of coefficients. Here, the proposed convex GWO/LMS adaptive filter guaranteed the same ERLE level regardless of the number of coefficients, as shown in Figure 5. Therefore, we used the minimum number of coefficients, since this factor is relevant, especially when it is implemented in resource-constrained devices.

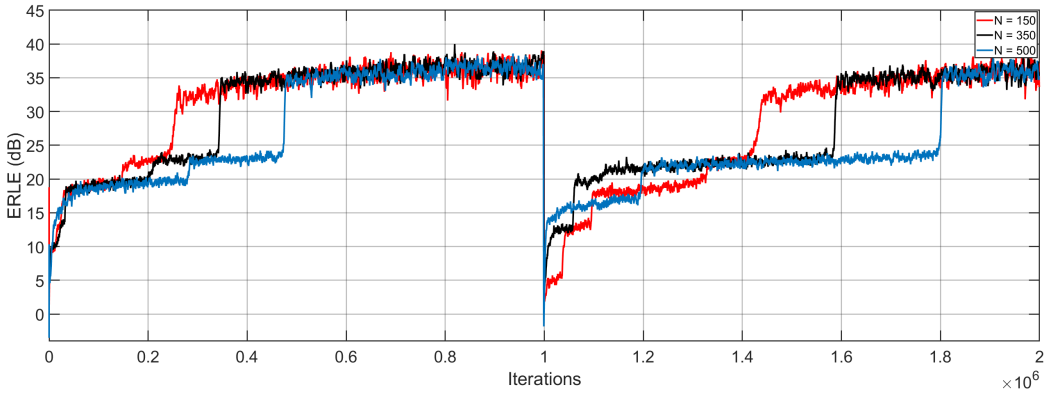


Figure 5. ERLE level for different number of coefficients N.

- *Effect of varying the number of search agents of the proposed convex GWO/LMS adaptive filter*. In this experiment, we varied the number of search agents from 30 to 200 to evaluate the performance of the proposed algorithm in terms of ERLE level. Since the minimum number of adaptive filter coefficients (150) guaranteed a good ERLE level, we used this number for the experiment. As can be observed from Figure 6, we obtained the same performance by using different numbers of search agents. In this way, we confirmed that, when employing the minimum number of search agents, the proposed method reached a good ERLE level. From an engineering perspective, this has a great impact on the performance of resource-constrained devices, since the proposed algorithm intends to reduce its computational cost by decreasing the number of search agents over the adaptive process.

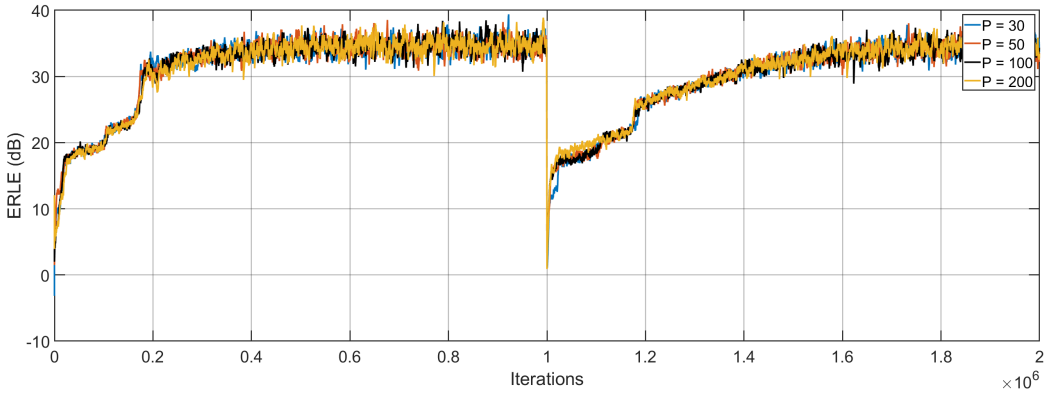


Figure 6. ERLE for different numbers of search agents P.

- *Performance comparison between the proposed convex GWO/LMS and existing approaches.* We performed two experiments to make a coherent comparison between the proposed convex GWO/LMS and the following existing approaches: LMS algorithm [1], conventional GWO [6], PSO [23], differential evolution (DE) algorithm [47], artificial bee colony optimization (ABC) [48], hybrid PSO–LMS [49] and modified ABC (MABC) [50]. In the first experiment, we shifted the acoustic path, and in the second experiment, we multiplied the acoustic path by -1 at the middle of the adaptive process. In addition, the tuning parameters of all the algorithms were selected to guarantee the best performance. Such tuning parameters are displayed in the following list:

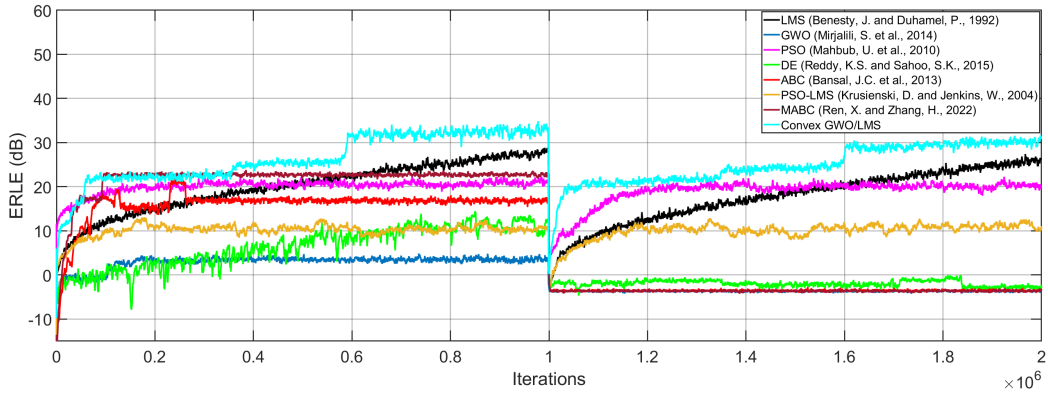
  1. **LMS**
     - Convergence factor $= 9 \times 10^{-7}$
  2. **GWO**
     - a decreases linearly from 2 to 0
     - lower bound $= -1$
     - Upper bound $= 1$
     - Population size $= 50$
  3. **PSO**
     - Acceleration coefficient, $c_1 = 1.6$
     - Acceleration coefficient, $c_2 = 1$
     - Inertia weight $= 0.8$
     - Lower bound $= -1$
     - Upper bound $= 1$
     - Population size $= 100$
  4. **DE**
     - Crossover rate $= 0.35$
     - Scaling factor $= 0.8$
     - Combination factor $= 0.25$
     - Lower bound $= -1$
     - Upper bound $= 1$
     - Population size $= 50$
  5. **ABC**
     - Evaporation parameter $= 0.1$
     - Pheromone $= 0.6$
     - Lower bound $= -1$
     - Upper bound $= 1$
     - Population size $= 50$

6. **PSO-LMS**
   - Acceleration coefficient, $c_1 = 0.00005$
   - Acceleration coefficient, $c_2 = 1.2$
   - Inertia weight $= 1$
   - Lower bound $= -1$
   - Upper bound $= 1$
   - Convergence factor $= 1 \times 10^{-9}$
   - Population size $= 60$
7. **MABC**
   - Evaporation parameter $= 0.1$
   - Pheromone $= 0.6$
   - Lower bound $= -1$
   - Upper bound $= 1$
   - Population size $= 50$
   - Convergence factor $= 3 \times 10^{-5}$

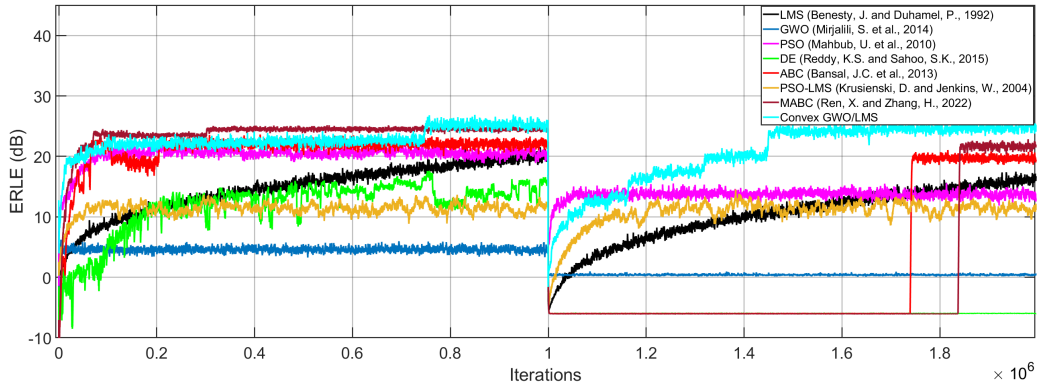
As can be observed from Figure 7, the proposed convex GWO/LMS adaptive filter showed the best performance, in terms of ERLE level and convergence speed, by expending a large number of additions and multiplications, as shown in Table 1. In contrast, the LMS algorithm expended fewer additions and multiplications compared with the proposed algorithm at the cost of exhibiting a slow convergence speed. In general, the excessive number of additions and multiplications makes the implementation of the GWO adaptive filter in current embedded devices, such as DSP and FPGA devices, impractical since they have a limited number of these circuits. Here, our proposal intended to dynamically decrease the number of search agents, as shown in Figure 8. As a consequence, the number of multiplications and additions also reduced (Equation (19)). In this way, the implementation of our proposal in embedded devices can be feasible.

Table 1. Comparison between the proposed convex GWO/LMS system and existing approaches in terms of the number of additions and multiplications.

Algorithm	Multiplications	Additions
LMS [1]	6,000,118,333	6,000,118,333
GWO [6]	1,349,996,758,333	2,249,994,508,333
PSO [23]	1,500,036,249,900	1,500,036,249,900
DE [47]	149,999,625,000	299,999,250,000
ABC [48]	600,058,499,850	749,938,125,150
PSO-LMS [49]	903,077,742,300	906,037,734,900
MABC [50]	1,799,955,500,100	1,620,075,949,800
Convex GWO/LMS	73,599,043,327	46,560,966,659



(**a**)



(**b**)

Figure 7. ERLE learning curves obtained by simulating existing approaches and the proposed algorithm; (**a**) by shifting the acoustic path at the middle of iterations and (**b**) by multiplying the acoustic path by −1 at the middle of iterations [1,15,23,48–51].

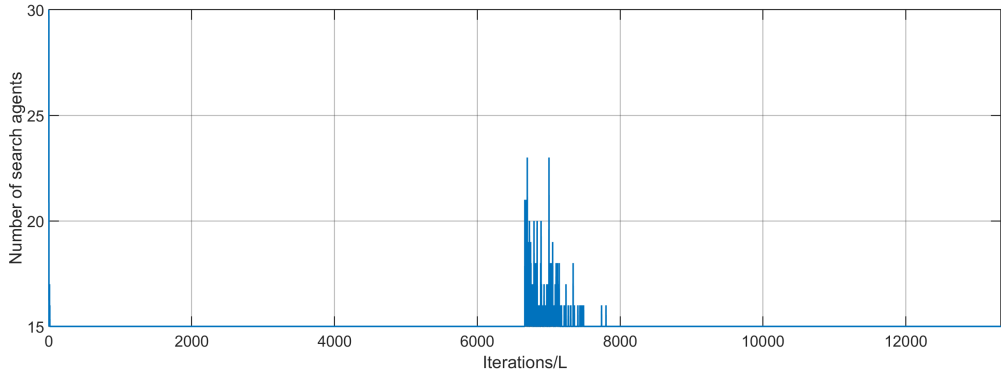


Figure 8. The number of search agents used during the adaptation process.

- *Statistical comparison between the proposed convex GWO/LMS and existing approaches* Statistical results were obtained with two different evaluations: average value of ERLE in dB and its corresponding standard deviation. The maximum number of iterations was set to 2,000,000 and each algorithm ran 10 times. The results are reported in Table 2.

Table 2. Comparison between the proposed convex GWO/LMS system and existing approaches in terms of average value of ERLE and standard deviation in dB.

Algorithm	Average Value	Standard Deviation
LMS [1]	14.7202	4.1761
GWO [6]	4.5653	0.3892
PSO [23]	20.2452	1.5622
DE [47]	11.7972	4.4434
ABC [48]	21.0656	3.4164
PSO-LMS [49]	11.2859	1.1497
MABC [50]	23.6927	3.0326
Convex GWO/LMS	22.7590	2.0259

As can be observed from Table 2, the proposed convex GWO/LMS achieved a good average ERLE level, in comparison with other existing algorithms. It should be noted that the MABC algorithm possessed the highest average value. Nonetheless, this algorithm presented a lower convergence speed, especially when abrupt changes occurred, as shown in Figure 7b. On the other hand, the GWO, PSO and PSO-LMS algorithms presented lower standard deviations in comparison with the proposed Convex GWO/LMS algorithm. Nonetheless, the proposed method achieved a higher average value.

4. Pure Hardware Simulation

To adequately simulate the proposed convex GWO/LMS system, we made extraordinary efforts to develop compact, high-processing and high-precision neural arithmetic circuits, such as adder and multiplier, since these two circuits are highly demanded in the computation of the proposed algorithm, as shown in Table 1. Specifically, we developed, for the first time, a customized floating-point representation to perform additions and multiplications with high precision.

To compute the numbers in floating-point format, we established the format criteria of the input numbers u and v to be either added or multiplier, as follows:

1. In general terms, the numbers u and v are separated in integer and fractional digits. Specifically, the number of integer digits and the number of fractional digits can be chosen over a range ($u_{g_0} \cdots u_{g_1} \cdots u_{g_m}$, $v_{g_0} \cdots v_{g_1} \cdots v_{g_m}$), as shown in Figure 9. Here, we painted the digits with a specific color by using a variant of the SN P systems called coloured spikes to easily distinguish the units, tens, hundreds, etc., of the integer part and tenths, hundredths, thousandths, etc., of the fractional part. This strategy was also used to represent the digits of the results of the addition and multiplication operations.

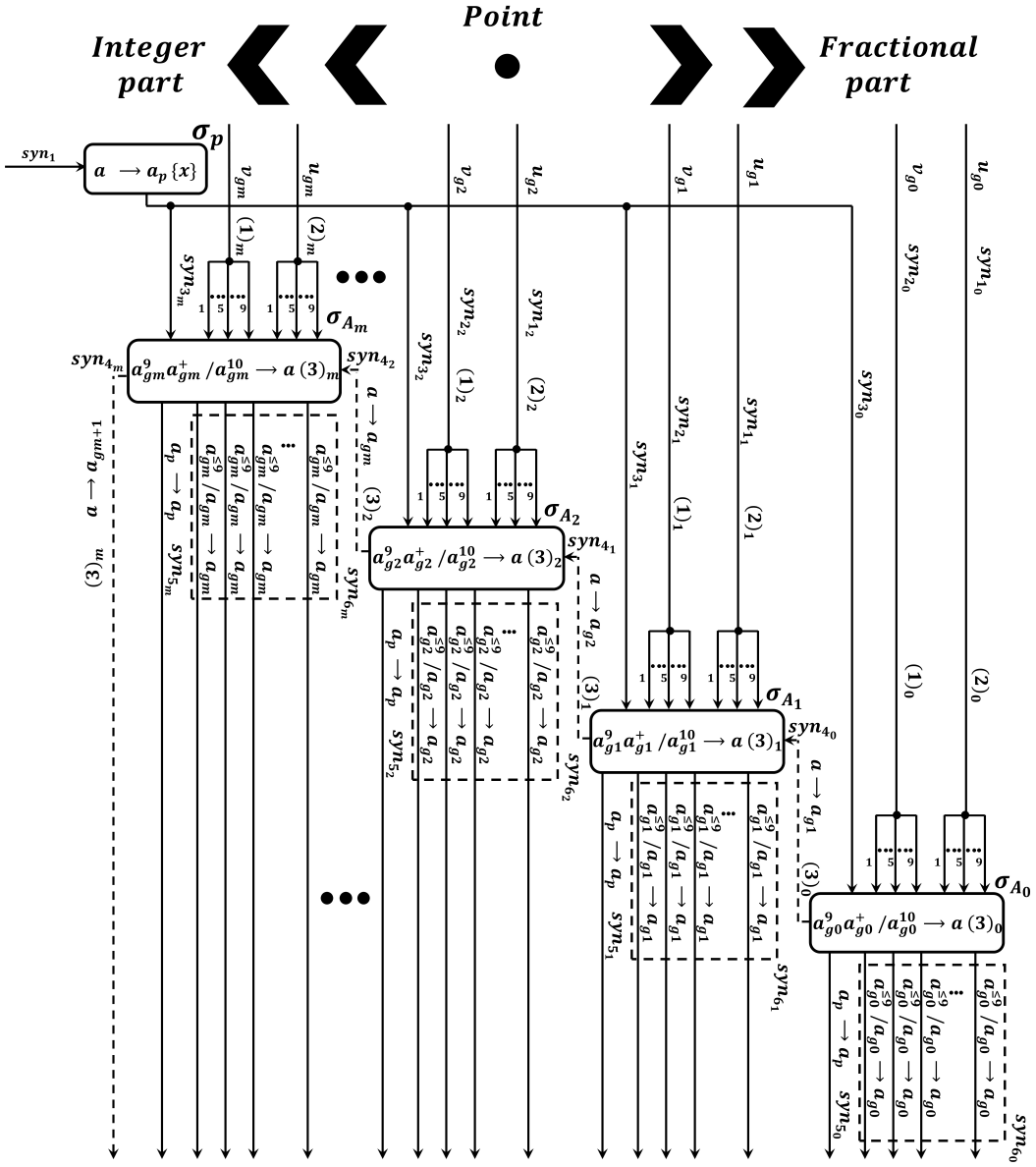


Figure 9. General structure of the proposed floating-point neural adder.

2. Here, each synaptic channel (1) and synaptic channel (2) has a set of dendritic branches. To perform customized floating-point addition, the integer and fractional digits of u and v were represented as the number of active dendritic branches, labeled as $1, \cdots, 9$. In the case of calculating a customized floating-point multiplication, the integer and fractional digits of u are represented as the number of spikes denoted in the extended rule $(a^p_u)^+/a^p_u \rightarrow a^p_u$. On the other hand, the integer and fractional digits of v activate the number of branches according to their value.

- *Parallel neural adder circuit* $\prod_{add}$.
  The proposed neural adder circuit $\prod_{add}$ has a set of neurons, $\sigma_{A_0}, \cdots, \sigma_{A_m}, \cdots$, and a neuron, σ_p. The set of neurons σ_A is in charge of computing the addition of two numbers, where each number is composed of an integer part and a fractional part, and neuron, σ_p, determines the position of the point to segment the number into an integer part and a fractional part, as shown in Figure 9.
  The proposed neural adder circuit $\prod_{add}$ computes the addition as follows:
  In the initial state, the neurons, σ_A, are empty. At this time, the dendritic branches of synaptic channels, (1) and (2), are activated according to the value of the digits, u and

v, respectively. For example, if the value of a digit, v or u, is equal to five, then five dendritic branches are activated. Therefore, these dendritic branches allow the flow of five spikes towards a specific neuron, σ_A. Simultaneously, the neuron, σ_p, places the point to segment the number into integer digits and the fractional digits by setting the firing rule, $a \rightarrow a_p\{X\}$. This spiking rule implies that, if neuron σ_p receives a spike at any time, it fires and sends a spike to a specific neuron, σ_{A_x}. To do this, we used a variant of the SN P systems called target indications. In this way, the point was allocated according to the desired precision. Therefore, the point, which was represented as a spike, was stored in a specific neuron, σ_{A_x}. Once the neural circuit was configured, the partial additions of the spikes started.

Here, the addition of the numbers was performed in a single simulation step, since all neurons, σ_A, processed their respective input spikes simultaneously. Additionally, the carry spike was obtained when any neuron, σ_A, accumulated ten spikes. At this moment, the spiking rule, $a^9a^+/a^{10} \rightarrow a(3)$, was set. After one simulation step, the result, represented by the remaining spikes in the soma of each neuron, is placed at the output synapses. In addition, neuron σ_{A_x} sends the spike point to the environment by enabling its spiking rule, $a_p \rightarrow a_p$.

In this work, we proved that the use of several variants of the SN P systems, such as coloured-spikes [36], rules on the synapses [37], target indications [51], extended channel rules [38], extended rules [39] and dendritic trunks [52] creates an ultra-compact and high performance circuit, instead of only using the soma as conventional SN P systems do. In particular, the proposed neural circuit required fewer simulation steps, neurons and synapses compared with the existing approach [34], as shown in Table 3.

Table 3. Comparison between the existing neural adder [34] and this work in terms of synapses/neurons and simulation steps. Here, n represents the number of digits of v and u. Adapted from [53].

Approach	[34]	This Work
Synapses	41 n	14 n
Neurons	12 n	n
Simulation steps	$28 + (n/2 - 1)$	1

- *Parallel neural multiplier circuit* $\prod_{mul}$.

  Since the multiplier is one of the most demanding in terms of processing and area consumption, a large number of techniques have been developed to minimize these factors. Recently, several authors have used SN P systems to create an efficient parallel multiplier circuit. However, the improvement of processing speed is still an issue since most of the studies improved the area consumption. The improvement of this factor potentially allows the development of high performance systems to support AEC systems in real-time. In addition, the development of a high-precision neural multiplier is still a challenging task, since this factor is especially relevant when metaheuristic algorithms are simulated. Here, we developed a neural multiplier that shows higher processing speeds, in comparison with existing approaches, by keeping the area consumption low. To achieve this, we reduced the processing time by using cutting-edge variants of the SN P systems, such as coloured-spikes [36], rules on the synapses [37], target indications [51], extended rules [39] and dendritic trunks [52]. Specifically, we used these variants to significantly improve the time expended in the computation of the partial products of the multiplication, in comparison with the most recent approach [35].

  The proposed neural multiplier circuit $\prod_{mul}$ is composed of a set of neurons (σ_{A_0}, $\cdots$, $\sigma_{A_{n-j}}$, $\cdots$, $\sigma_{A_{n-1}}$, $\cdots$, $\sigma_{A_{2(n-j)}}$, $\cdots$, $\sigma_{A_{2n-1}}$), and neuron, σ_{in}, as shown in Figure 10. In general terms, neurons, σ_A, perform the addition of the partial products, where

each partial product is computed by using dendritic branches. In particular, each neuron σ_A computes the partial product between a single digit of u and a digit of v, where the digits of u are represented as p spikes, which are generated by neuron σ_{in} when its spiking rule, $(a_u^p)^+ / a_u^p \rightarrow a_u^p$, is applied, and the value of each digit of v activates an equal number of dendritic branches.

The proposed neural multiplier circuit $\prod_{mul}$ performs the multiplication, as follows: At the initial simulation step, neurons, σ_A, are empty. At this time, neuron, σ_{in}, places the spike point by receiving a spike. Therefore, it fires and sends the spike point to a specific neuron, σ_{A_x}, using the target indications [51]. In this way, the digits of u or v are segmented into integer and fractional parts. Once the multiplier is configured, $m \cdot n$ partial products are executed in parallel. To perform any partial product, neuron, σ_{in}, fires p spikes which are sent to its corresponding neuron, σ_A. The soma of this neuron receives many copies of the spikes, p, in function of the number of active dendritic branches. For example, if neuron multiplies 3×3, this implies that neuron, σ_{A_x}, receives three copies of three spikes by means of three dendritic branches. In this way, the neuron, σ_A, increases its potential of the soma by performing three additions. Therefore, the synaptic weights are not required since many approaches use them to perform partial products. Hence, the number of branched connections can be variable. In this way, the neuron σ_A enables the optimal number of synaptic connections. From an engineering perspective, we proposed the use of a variable number of forked connections, since the implementation of a very-large number of synaptic connections in advanced FPGAs creates critical routing problems. Once the result is obtained, neuron, σ_{in}, places the spike point according to the addition of the number of fractional digits, as in conventional multiplication. Therefore, σ_A, which received the spike point at the initial simulation, fired the point spike by applying it firing rule ($a_p \rightarrow a_p$).

As can be observed from Table 4, we achieved a significant improvement in terms of simulation steps, since only one simulation step was required to perform a multiplication of two numbers with any length. This aspect is relevant, especially when real-time AEC system simulations are required. Additionally, we reduced the number of synapses in comparison with the existing work.

Table 4. Comparison between the proposed neural multiplier and the existing neural multiplier [35] in terms of simulation steps, synapses and neurons. Where n denotes the number of digits of v and u. Adapted from [53].

Existing Neural Multiplier	**[35]**	**This Work**
Synapses	$9 \cdot n \cdot m$	$n \cdot m$
Neurons	n	n
Simulation steps	10	1

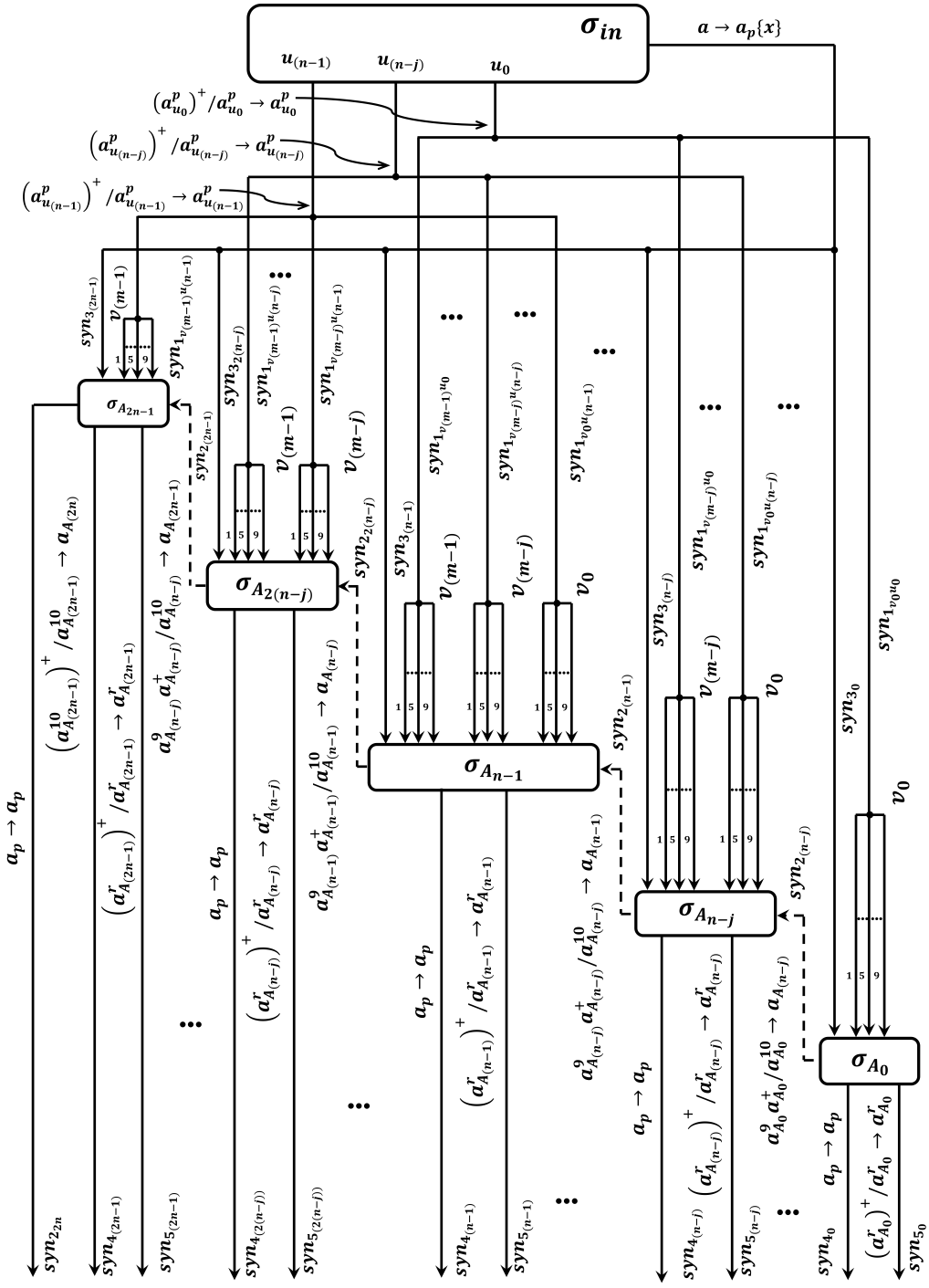


Figure 10. Structure of the neural multiplier.

4.1. Experimental Results

To demonstrate the computational capabilities of the proposed convex GWO/LMS adaptive filter, we considered an arbitrary single-talk scenario, as shown in Figure 11.

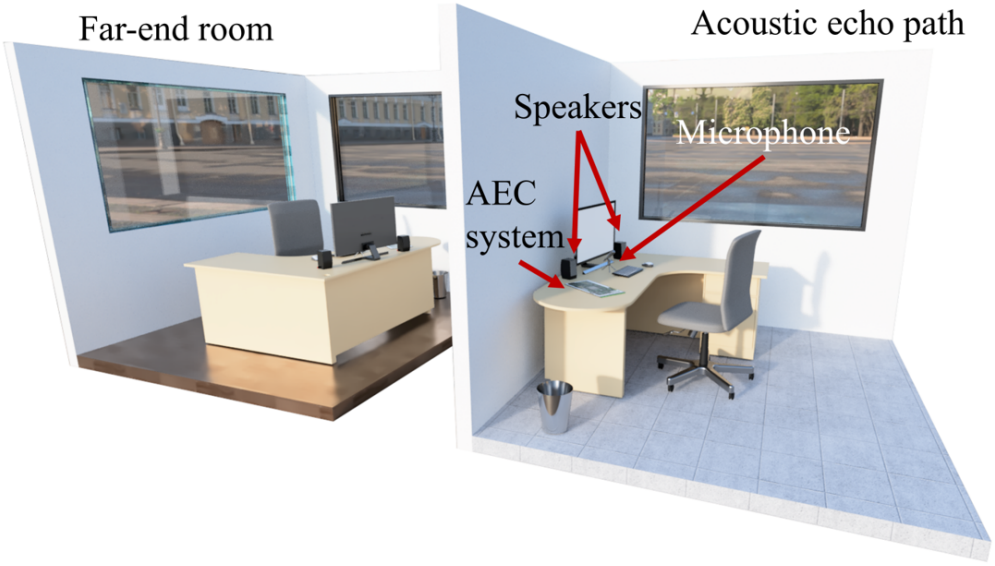


Figure 11. Scheme of the AEC prototype.

Under this configuration, we used the proposed AEC system to perform the simulation of the proposed algorithm by using the experimental setup of Figure 11. Specifically, the proposed system has an AEC neural processor as its main core to cancel the echo signal and the background noise by simulating the proposed convex GWO/LMS adaptive filter. As can be observed from Figure 12, the AEC neural processor is composed of a control unit, CU, a set of processing cores, α, β, δ, ω, LMS, convex, and BRAMs. Figure 13 shows a sequence diagram to specify how each processing core computes specific parts of the proposed convex GWO/LMS adaptive filter. Here, we used the BRAMs to store L and N samples of the signals, x, and d, where $L = N$. In this way, we implemented the block processing scheme. Once these values are stored in their respective BRAMs, the computation of the proposed convex GWO/LMS adaptive filter starts. In this way, the AEC neural processor computes the new variant of the GWO algorithm and the LMS algorithm simultaneously. Specifically, we propose a new time-multiplexing control scheme to automatically update the number of search agents of the GWO algorithm over the processing time. Under this scheme, the number of search agents, ω, which are divided into three parts, is enabled or disabled either by the processor, α, or β, or δ. It is important to keep in mind that processing cores, α, β, δ evaluate the response of their respective processors, ω, simultaneously. According to this, the proposed time-multiplexing control scheme uses the signals, en_w_α, en_w_β, and en_w_δ, to enable or disable the number of search agents according to the simulation needs, as shown in Figure 12. Here, the use of the time-multiplexing control scheme allowed us to significantly decrease the number of buses to transfer the coefficients between the processing cores, ω, and the processing cores, α, or β, or δ. This saving allowed us to implement a full connection between the processing cores, α, β and δ. As a consequence, the evaluation of the searching point of each agent was performed by processing cores, α, β and δ simultaneously.

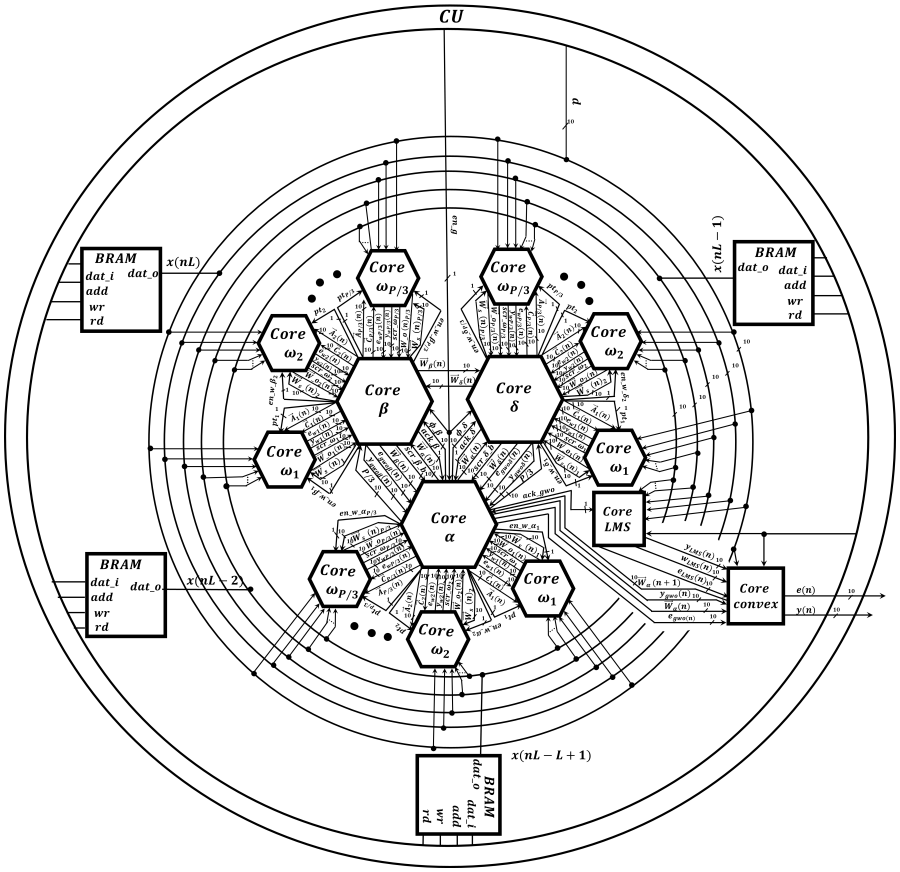


Figure 12. Scheme of the proposed AEC neural processor.

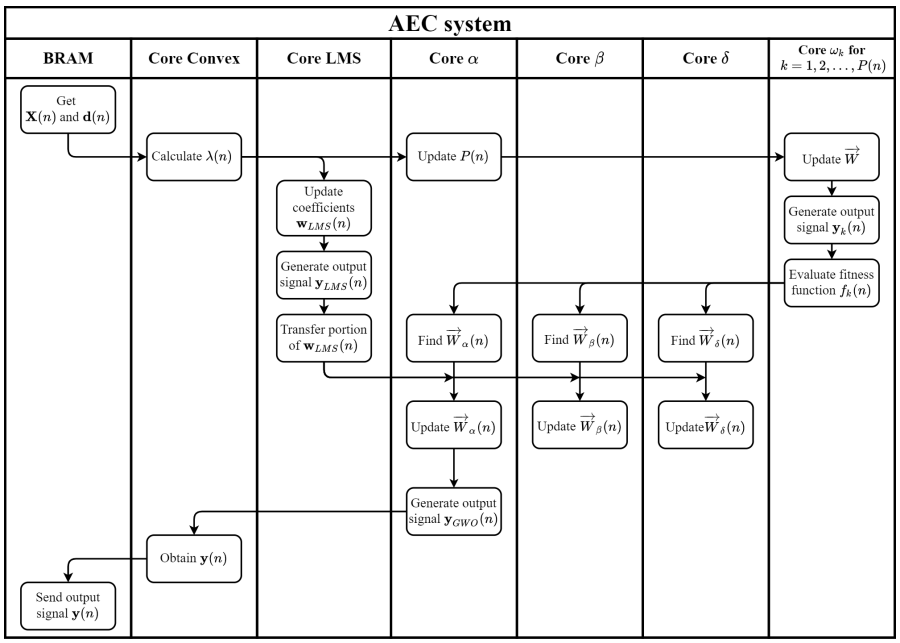


Figure 13. The sequential diagram to describe how the proposed convex GWO/LMS adaptive filter is executed by means of the processing cores.

In this work, we used the proposed AEC neural processor to simulate the single-talk and double-talk scenarios by considering the following conditions:

- We employed 1024 adaptive filter coefficients and varied the search agents from 70 to 15.

- The filter and the block had the same length as the echo path.
- As input signals, we used an AR(1) process and speech sequence signals.
- The step-size of the LMS algorithm was set to $\mu = 0.000001$ and the step-size of the convex algorithm was selected to be $\mu_a = 15$
- The search agents were initialized using normally distributed random numbers and their positions were bounded between $[-1, 1]$ over the filtering process.

4.2. Single-Talk Scenario

In this section, we demonstrate the computational capabilities of the proposed algorithm under a single-talk scenario. As can be observed from Figures 14 and 15, the proposed AEC neural processor, which simulated the proposed convex GWO/LMS algorithm, reached a good ERLE level by processing two different signals. To carry this out, we configured the AEC neural processor to support one α, one β, one δ, and 70 ω processing cores. Here, the most demanding core, in terms of area and processing speed, was the ω processing core, since each one required 1000 neural multipliers and 1000 neural adders. Therefore, we physically implemented six ω processing cores to simulate virtually 70 ω processing cores. In this way, we saved a large area consumption. In general terms, the implementation of these components required 420,380 LEs, which represented 79% of the total area of an Stratix IV GX EP4SGX530 FPGA. Furthermore, the AEC neural processor required 114.56 µs, which was obtained by multiplying 14,320 clock cycles by the system clock period (8 ns) to simulate the proposed convex GWO/LMS algorithm. This time was calculated when all the search agents were used, i.e., by considering the worse case. However, the number of search agents decreased over the processing time. In the case of employing fifteen search agents, which represented the minimum number, only 1300 clock cycles were expended. Therefore, the processing time reduced from 114.56 µs to 10.4 µs. Therefore, the simulation of real-time AEC systems was guaranteed since the maximum latency was 125 µs.

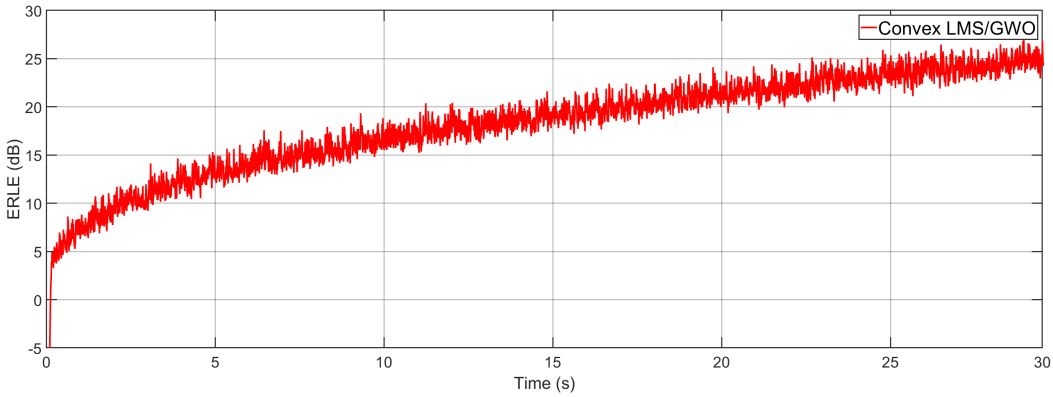


Figure 14. ERLE learning curve of the proposed convex GWO/LMS considering an AR(1) process as input signal.

4.3. Double-Talk Scenario

To perform the experiments under this configuration, we employed a double-talk detector circuit to avoid adaptation during periods of simultaneous far and near-end speech. It should be noted that the area consumption in the implementation of this circuit was negligible. Therefore, this implementation expended around 79% of the total area of an Stratix IV GX EP4SGX530 FPGA, as in the previous case. Figures 16 and 17 show the ERLE of the proposed Convex GWO/LMS algorithm by considering an AR(1) process and a speech sequence signal, respectively. As can be observed from the above figures, the proposed algorithm showed good tracking capabilities and achieved a good ERLE level. This aspect is relevant since these levels are required in the development of practical and real-world AEC applications.

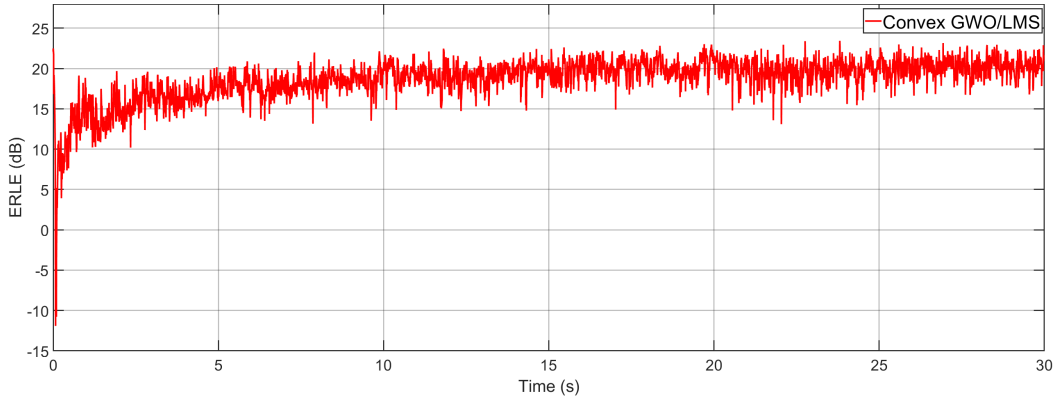


Figure 15. ERLE learning curve of the proposed convex GWO/LMS considering a speech sequence signal as input signal.

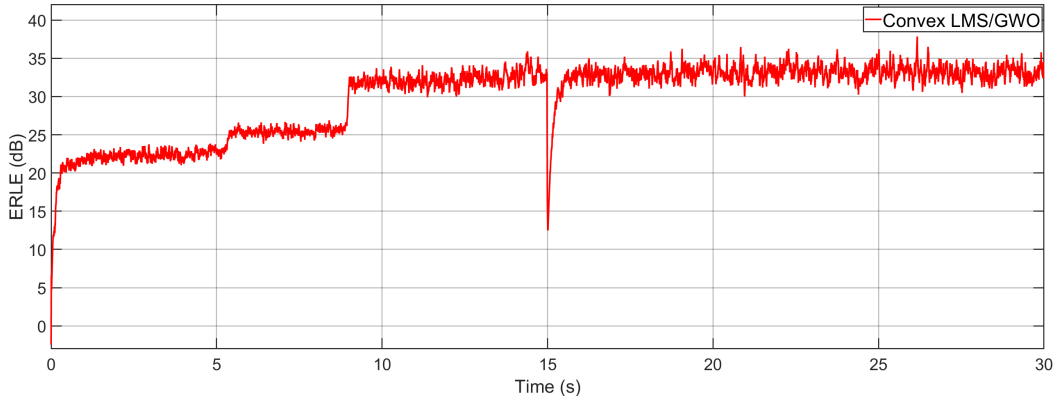


Figure 16. ERLE learning curve of the proposed convex GWO/LMS considering an AR(1) process as input signal and a double-talk scenario.

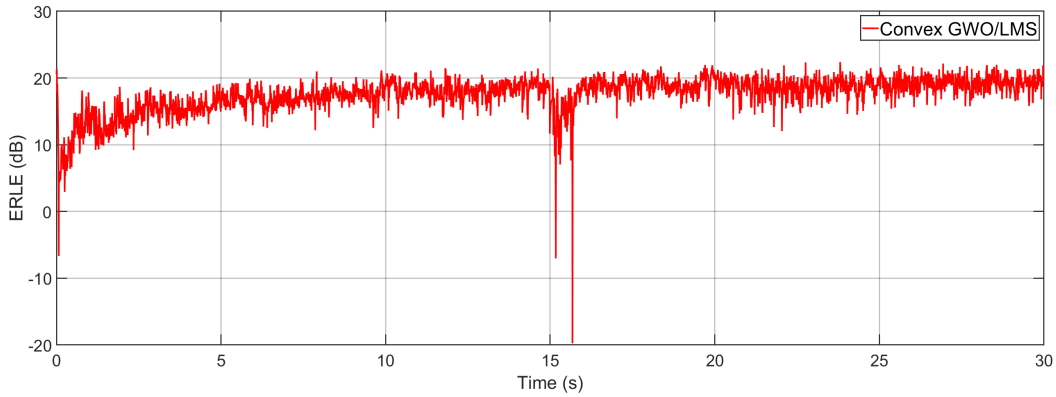


Figure 17. ERLE learning curve of the proposed convex GWO/LMS considering a speech sequence signal as input signal considering a double-talk scenario.

5. Conclusions

In this work, we present, for the first time, the development of a high-speed and compact FPGA-based AEC neural processor to efficiently simulate a new convex GWO/LMS algorithm. Here, we grouped our contributions as follows:

- *From the AEC model point of view*.
  Here, we made intensive efforts to reduce the computational cost of the AEC systems to be implemented in resource-constrained devices. In addition, we significantly increased the convergence properties of these systems by using a cutting-edge metaheuristic swarm intelligence method, in combination with a gradient descent algorithm to be used in practical acoustic environments. Specifically, we present a new

variant of GWO algorithm along with the LMS algorithm. The use of this combination allowed us to guarantee a higher convergence rate and lower MSE level, in comparison to when gradient descent algorithms or metaheuristic SI methods were used separately. To improve the tracking capabilities of the conventional GWO algorithm, the proposed variant has new exploration capabilities, since the search space is dynamically adjusted. To make the implementation of the proposed variant of the GWO algorithm in embedded devices feasible, we used the block-processing scheme. In this way, the proposed convex GWO/LMS algorithm can be easily implemented in parallel hardware architectures. As a consequence, it can be simulated at high processing speeds. In addition, we significantly reduced the computational cost of the proposed convex GWO/LMS algorithm. To achieve this aim, we propose a method to dynamically decrease the population of a variant of the GWO algorithm over the filtering process.

- *From the SN P systems point of view*.
  Here, we present, for the first time, a compact and high-processing speed floating-point neural adder and multiplier circuit. We used cutting-edge variants of the SN P systems, coloured-spikes, rules on the synapses, target indications, extended channel rules, extended rules and dendritic trunks to create a customized floating-point neural adder and multiplier. Specifically, the proposed neural adder and multiplier exhibits higher processing speed, compared with existing SN P adders and multipliers, since both expend only one simulation step, which is the best improvement achieved until now.
- *From the digital point of view*.
  In this work, we present, for the first time, the development of a parallel hardware architecture to simulate a variable number of search agents by using the proposed time-multiplexing control scheme. In this way, we implemented the proposed GWO method properly, in which the number of search agents increase or decrease according to the simulation needs. In addition, the use of this scheme allowed us to exploit, to the maximum, the flexibility and scalability features of the GWO algorithm.

Finally, we carried out several experiments to prove that the proposed convex GWO/LMS algorithm, along with the new techniques inspired by biological neural processes, potentially allow the creation of practical and real-time AEC processing tools. Part of the future work is to develop new convex combinations, in which other meta-heuristic algorithms can be used in other adaptive filtering applications, such as active noise control, channel equalization and noise cancellers. In addition, new digital techniques will be explored to mimic bio-inspired behavior with high accuracy.

Author Contributions: Conceptualization, E.P.; Data curation, L.G. and H.M.P.; Formal analysis, G.S.; Funding acquisition, J.G.A. and J.C.S.; Investigation, A.V. and J.G.A.; Methodology, J.G.A. and G.S.; Resources, E.P., E.A. and A.V.; Software, E.P., E.A. and A.V.; Supervision, G.S. and J.C.S.; Validation, E.P. and E.A.; Writing—original draft, L.G. and H.M.P.; Writing—review & editing, H.M.P. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to thank the Instituto Politécnico Nacional for the financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: The authors would like to thank the Consejo Nacional de Ciencia y Tecnologia (CONACYT) and the IPN for the financial support to make this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Benesty, J.; Duhamel, P. A fast exact least mean square adaptive algorithm. *IEEE Trans. Signal Process.* **1992**, *40*, 2904–2920. [CrossRef] [PubMed]
2. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the MHS'95, Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
3. Ling, Q.; Ikbal, M.A.; Kumar, P. Optimized LMS algorithm for system identification and noise cancellation. *J. Intell. Syst.* **2021**, *30*, 487–498. [CrossRef]
4. Botzheim, J.; Cabrita, C.; Kóczy, L.T.; Ruano, A. Fuzzy rule extraction by bacterial memetic algorithms. *Int. J. Intell. Syst.* **2009**, *24*, 312–339. [CrossRef]
5. Ariyarit, A.; Kanazaki, M. Multi-modal distribution crossover method based on two crossing segments bounded by selected parents applied to multi-objective design optimization. *J. Mech. Sci. Technol.* **2015**, *29*, 1443–1448. [CrossRef]
6. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
7. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
8. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
9. Khehra, B.S.; Singh, A.; Kaur, L.M. Masi Entropy-and Grey Wolf Optimizer-Based Multilevel Thresholding Approach for Image Segmentation. *J. Inst. Eng. Ser. B* **2022**, *103*, 1619–1642. [CrossRef]
10. Vashishtha, G.; Kumar, R. An amended grey wolf optimization with mutation strategy to diagnose bucket defects in Pelton wheel. *Measurement* **2022**, *187*, 110272. [CrossRef]
11. Rajammal, R.R.; Mirjalili, S.; Ekambaram, G.; Palanisamy, N. Binary Grey Wolf Optimizer with Mutation and Adaptive K-nearest Neighbour for Feature Selection in Parkinson's Disease Diagnosis. *Knowl.-Based Syst.* **2022**, *246*, 108701. [CrossRef]
12. Reddy, V.P.C.; Gurrala, K.K. Joint DR-DME classification using deep learning-CNN based modified grey-wolf optimizer with variable weights. *Biomed. Signal Process. Control* **2022**, *73*, 103439. [CrossRef]
13. Dey, S.; Banerjee, S.; Dey, J. Implementation of Optimized PID Controllers in Real Time for Magnetic Levitation System. In *Computational Intelligence in Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 249–256.
14. Zhang, X.; Li, D.; Li, J.; Liu, B.; Jiang, Q.; Wang, J. Signal-Noise Identification for Wide Field Electromagnetic Method Data Using Multi-Domain Features and IGWO-SVM. *Fractal Fract.* **2022**, *6*, 80. [CrossRef]
15. Premkumar, M.; Jangir, P.; Kumar, B.S.; Alqudah, M.A.; Nisar, K.S. Multi-objective grey wolf optimization algorithm for solving real-world BLDC motor design problem. *Comput. Mater. Contin.* **2022**, *70*, 2435–2452. [CrossRef]
16. Nagadurga, T.; Narasimham, P.; Vakula, V.; Devarapalli, R. Gray wolf optimization-based optimal grid connected solar photovoltaic system with enhanced power quality features. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6696. [CrossRef]
17. Musharavati, F.; Khoshnevisan, A.; Alirahmi, S.M.; Ahmadi, P.; Khanmohammadi, S. Multi-objective optimization of a biomass gasification to generate electricity and desalinated water using Grey Wolf Optimizer and artificial neural network. *Chemosphere* **2022**, *287*, 131980. [CrossRef]
18. Meidani, K.; Hemmasian, A.; Mirjalili, S.; Barati Farimani, A. Adaptive grey wolf optimizer. *Neural Comput. Appl.* **2022**, *34*, 7711–7731. [CrossRef]
19. Zhang, L.; Yu, C.; Tan, Y. A method for pulse signal denoising based on VMD parameter optimization and Grey Wolf optimizer. Journal of Physics: Conference Series. In Proceedings of the 2021 2nd International Conference on Electrical, Electronic Information and Communication Engineering (EEICE 2021), Tianjin, China, 16–18 April 2021; Volume 1920, p. 012100.
20. Negi, G.; Kumar, A.; Pant, S.; Ram, M. GWO: A review and applications. *Int. J. Syst. Assur. Eng. Manag.* **2021**, *12*, 1–8. [CrossRef]
21. Faris, H.; Aljarah, I.; Al-Betar, M.A.; Mirjalili, S. Grey wolf optimizer: A review of recent variants and applications. *Neural Comput. Appl.* **2018**, *30*, 413–435. [CrossRef]
22. Salinas, G.; Pichardo, E.; Vázquez, Á.A.; Avalos, J.G.; Sánchez, G. Grey wolf optimization algorithm for embedded adaptive filtering applications. *IEEE Embed. Syst. Lett.* **2022**, 1. [CrossRef]
23. Mahbub, U.; Acharjee, P.P.; Fattah, S.A. A time domain approach of acoustic echo cancellation based on particle swarm optimization. In Proceedings of the International Conference on Electrical & Computer Engineering (ICECE 2010), Dhaka, Bangladesh, 18–20 December 2010; pp. 518–521.
24. Mahbub, U.; Acharjee, P.P.; Fattah, S.A. An acoustic echo cancellation scheme based on particle swarm optimization algorithm. In Proceedings of the TENCON 2010—2010 IEEE Region 10 Conference, Fukuoka, Japan, 21–24 November 2010; pp. 759–762.
25. Kimoto, M.; Asami, T. Multichannel Acoustic Echo Canceler Based on Particle Swarm Optimization. *Electron. Commun. Jpn.* **2016**, *99*, 31–40. [CrossRef]
26. Mishra, A.K.; Das, S.R.; Ray, P.K.; Mallick, R.K.; Mohanty, A.; Mishra, D.K. PSO-GWO optimized fractional order PID based hybrid shunt active power filter for power quality improvements. *IEEE Access* **2020**, *8*, 74497–74512. [CrossRef]
27. Suman, S.; Chatterjee, D.; Mohanty, R. Comparison of PSO and GWO Techniques for SHEPWM Inverters. In Proceedings of the 2020 International Conference on Computer, Electrical & Communication Engineering (ICCECE), Kolkata, India, 17–18 January 2020; pp. 1–7.
28. Şenel, F.A.; Gökçe, F.; Yüksel, A.S.; Yiğit, T. A novel hybrid PSO–GWO algorithm for optimization problems. *Eng. Comput.* **2019**, *35*, 1359–1373. [CrossRef]

29. Wang, W.; Wang, J. Convex combination of two geometric-algebra least mean square algorithms and its performance analysis. *Signal Process.* **2022**, *192*, 108333. [CrossRef]
30. Bakri, K.J.; Kuhn, E.V.; Matsuo, M.V.; Seara, R. On the behavior of a combination of adaptive filters operating with the NLMS algorithm in a nonstationary environment. *Signal Process.* **2022**, *196*, 108465. [CrossRef]
31. Jeong, J.J.; Kim, S. Robust adaptive filter algorithms against impulsive noise. *Circuits Syst. Signal Process.* **2019**, *38*, 5651–5664. [CrossRef]
32. Silva, M.T.; Nascimento, V.H. Improving the tracking capability of adaptive filters via convex combination. *IEEE Trans. Signal Process.* **2008**, *56*, 3137–3149. [CrossRef]
33. Ionescu, M.; Păun, G.; Yokomori, T. Spiking neural P systems. *Fundam. Inform.* **2006**, *71*, 279–308.
34. Frias, T.; Sanchez, G.; Garcia, L.; Abarca, M.; Diaz, C.; Sanchez, G.; Perez, H. A new scalable parallel adder based on spiking neural P systems, dendritic behavior, rules on the synapses and astrocyte-like control to compute multiple signed numbers. *Neurocomputing* **2018**, *319*, 176–187. [CrossRef]
35. Avalos, J.G.; Sanchez, G.; Trejo, C.; Garcia, L.; Pichardo, E.; Vazquez, A.; Anides, E.; Sanchez, J.C.; Perez, H. High-performance and ultra-compact spike-based architecture for real-time acoustic echo cancellation. *Appl. Soft Comput.* **2021**, *113*, 108037. [CrossRef]
36. Song, T.; Rodríguez-Patón, A.; Zheng, P.; Zeng, X. Spiking neural P systems with colored spikes. *IEEE Trans. Cogn. Dev. Syst.* **2017**, *10*, 1106–1115. [CrossRef]
37. Peng, H.; Chen, R.; Wang, J.; Song, X.; Wang, T.; Yang, F.; Sun, Z. Competitive spiking neural P systems with rules on synapses. *IEEE Trans. NanoBiosci.* **2017**, *16*, 888–895. [CrossRef]
38. Lv, Z.; Bao, T.; Zhou, N.; Peng, H.; Huang, X.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. Spiking neural p systems with extended channel rules. *Int. J. Neural Syst.* **2021**, *31*, 2050049. [CrossRef] [PubMed]
39. Chen, H.; Ionescu, M.; Ishdorj, T.O.; Păun, A.; Păun, G.; Pérez-Jiménez, M.J. Spiking neural P systems with extended rules: Universality and languages. *Nat. Comput.* **2008**, *7*, 147–166. [CrossRef]
40. Adam, S.P.; Alexandropoulos, S.A.N.; Pardalos, P.M.; Vrahatis, M.N. No free lunch theorem: A review. In *Approximation and Optimization*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 57–82.
41. Scarpiniti, M.; Comminiello, D.; Uncini, A. Convex combination of spline adaptive filters. In Proceedings of the 2019 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2–6 September 2019; pp. 1–5.
42. Khan, M.T.; Kumar, J.; Ahamed, S.R.; Faridi, J. Partial-LUT designs for low-complexity realization of DA-based BLMS adaptive filter. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *68*, 1188–1192. [CrossRef]
43. Khan, M.T.; Shaik, R.A. Analysis and implementation of block least mean square adaptive filter using offset binary coding. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
44. International Telecommunication Union ITU-T. *Digital Network Echo Cancellers*; Standardization Sector of ITU: Geneva, Switzerland, 2002.
45. Clark, G.; Mitra, S.; Parker, S. Block implementation of adaptive digital filters. *IEEE Trans. Acoust. Speech Signal Process.* **1981**, *29*, 744–752. [CrossRef]
46. Burrus, C. Block implementation of digital filters. *IEEE Trans. Circuit Theory* **1971**, *18*, 697–701. [CrossRef]
47. Reddy, K.S.; Sahoo, S.K. An approach for FIR filter coefficient optimization using differential evolution algorithm. *AEU-Int. J. Electron. Commun.* **2015**, *69*, 101–108. [CrossRef]
48. Bansal, J.C.; Sharma, H.; Jadon, S.S. Artificial bee colony algorithm: A survey. *Int. J. Adv. Intell. Paradig.* **2013**, *5*, 123–159. [CrossRef]
49. Krusienski, D.; Jenkins, W. A particle swarm optimization-least mean squares algorithm for adaptive filtering. In Proceedings of the Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 7–10 November 2004; Volume 1, pp. 241–245.
50. Ren, X.; Zhang, H. An Improved Artificial Bee Colony Algorithm for Model-Free Active Noise Control: Algorithm and Implementation. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–11. [CrossRef]
51. Wu, T.; Zhang, L.; Pan, L. Spiking neural P systems with target indications. *Theor. Comput. Sci.* **2021**, *862*, 250–261. [CrossRef]
52. Garcia, L.; Sanchez, G.; Vazquez, E.; Avalos, G.; Anides, E.; Nakano, M.; Sanchez, G.; Perez, H. Small universal spiking neural P systems with dendritic/axonal delays and dendritic trunk/feedback. *Neural Netw.* **2021**, *138*, 126–139. [CrossRef]
53. Maya, X.; Garcia, L.; Vazquez, A.; Pichardo, E.; Sanchez, J.C.; Perez, H.; Avalos, J.G.; Sanchez, G. A high-precision distributed neural processor for efficient computation of a new distributed FxSMAP-L algorithm applied to real-time active noise control systems. *Neurocomputing* **2023**, *518*, 545–561. [CrossRef]