micromachines

MDPI

Article

A Real-Time FPGA-Based Metaheuristic Processor to Efficiently Simulate a New Variant of the PSO Algorithm

Esteban Anides, Guillermo Salinas, Eduardo Pichardo, Juan G. Avalos, Giovanny Sánchez *, Juan C. Sánchez, Gabriel Sánchez, Eduardo Vazquez and Linda K. Toscano

Instituto Politécnico Nacional, ESIME Culhuacan, Av. Santa Ana No. 1000, Ciudad de México 04260, Mexico
* Correspondence: gsanchezriv@ipn.mx; Tel.: +52-55-1704-9527

Abstract: Nowadays, high-performance audio communication devices demand superior audio quality. To improve the audio quality, several authors have developed acoustic echo cancellers based on particle swarm optimization algorithms (PSO). However, its performance is reduced significantly since the PSO algorithm suffers from premature convergence. To overcome this issue, we propose a new variant of the PSO algorithm based on the Markovian switching technique. Furthermore, the proposed algorithm has a mechanism to dynamically adjust the population size over the filtering process. In this way, the proposed algorithm exhibits great performance by reducing its computational cost significantly. To adequately implement the proposed algorithm in a Stratix IV GX EP4SGX530 FPGA, we present for the first time, the development of a parallel metaheuristic processor, in which each processing core simulates the different number of particles by using the time-multiplexing technique. In this way, the variation of the size of the population can be effective. Therefore, the properties of the proposed algorithm along with the proposed parallel hardware architecture potentially allow the development of high-performance acoustic echo canceller (AEC) systems.

Keywords: FPGA; parallel metaheuristic processor; particle swarm optimization; Markovian switching technique; AEC system; spiking neural P systems

Citation: Anides, E.; Salinas, G.; Pichardo, E.; Avalos, J.G.; Sánchez, G.; Sánchez, J.C.; Sánchez, G.; Vazquez, E.; Toscano, L.K. A Real-Time FPGA-Based Metaheuristic Processor to Efficiently Simulate a New Variant of the PSO Algorithm. *Micromachines* **2023**, *14*, 809. https://doi.org/10.3390/mi14040809

Academic Editor: José de Jesús Rangel Magdaleno

Received: 17 February 2023
Revised: 24 March 2023
Accepted: 29 March 2023
Published: 31 March 2023

1. Introduction

Over the last ten years, tremendous efforts have been made to develop high-performance acoustic echo cancellers, which can offer high-quality and realistic sound needed in the newest acoustic communications. In particular, some authors have used the PSO algorithm as an adaptive echo canceling algorithm since this can be easily implemented and exhibits a fast convergence rate [1]. For example, Mahbub et al. [1,2] presented an AEC system based on the PSO to compute the error minimization in the frequency domain and time domain, respectively. Pichardo et al. [3] introduced a convex combination to improve the performance of the digital filter at the cost of increasing the overall computational cost. Recently, Kimoto et al. [4] introduced a multichannel adaptive echo-canceling algorithm based on PSO. Specifically, their technique considers a pre-processing of the input signals. Despite achieving these advanced approaches, there are still remaining tasks to significantly improve the performance of these systems in terms of echo return loss enhancement (ERLE) and convergence rate. Unfortunately, PSO suffers from premature convergence problems, especially in the case of multi-modal optimization problems. This reduces its performance since it losses the ability to find the optimal solution. To deal with this, several authors have proposed modifications to the conventional PSO [5–10]. However, a small number of solutions have been applied to adaptive acoustic echo cancellers. On the other hand, the implementation of the PSO algorithm in FPGA devices, to simulate AEC systems, faces great challenges to build optimal hardware architectures. To date, several hardware architectures have been developed to implement the PSO algorithm to be applied in adaptive filtering [11–13]. However, none of these architectures have been proposed to

simulate AEC systems. Here, we present for the first time, the implementation of the PSO algorithm in an FPGA for AEC systems. Specifically, we include the Markovian switching technique [14] into the conventional PSO algorithm to create a high-performance AEC system since considers quick convergence to the global optimum and also keeps the swarm global search simultaneously by taking advantage of the current search information.

From the engineering point of view, the implementation of the AEC systems based on the PSO algorithm requires a large number of particles. As a consequence, a large area of consumption is required. To overcome this, we proposed criteria to dynamically decrease the number of particles over the filtering process. In addition, we use the block-processing scheme to easily implement the proposed algorithm in a parallel hardware architecture.

2. Proposed Markov Switching PSO Algorithm

In this section, we present a new variant of the PSO to improve search performance. Specifically, we dynamically adjust the velocity of the particle according to an evolutionary factor. In this manner, the premature convergence of the PSO can be prevented and this can be also especially useful in dealing with multi-modal and high-dimensional problems. Figure 1 shows the structure of the proposed variant of the PSO algorithm applied to adaptive filtering and Figure 2 shows the required steps to perform the proposed PSO algorithm.

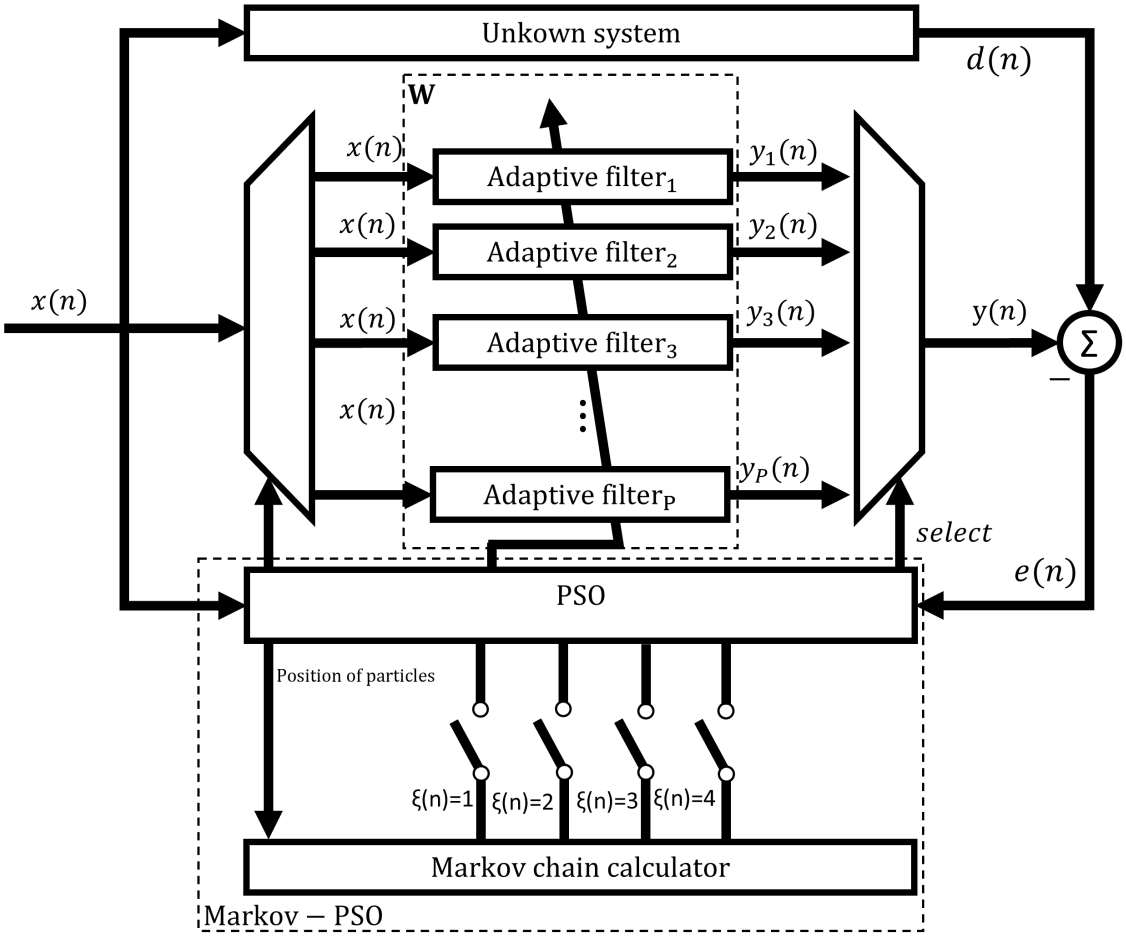

Figure 1. Structure of the proposed Markov switching PSO algorithm applied to adaptive filtering.

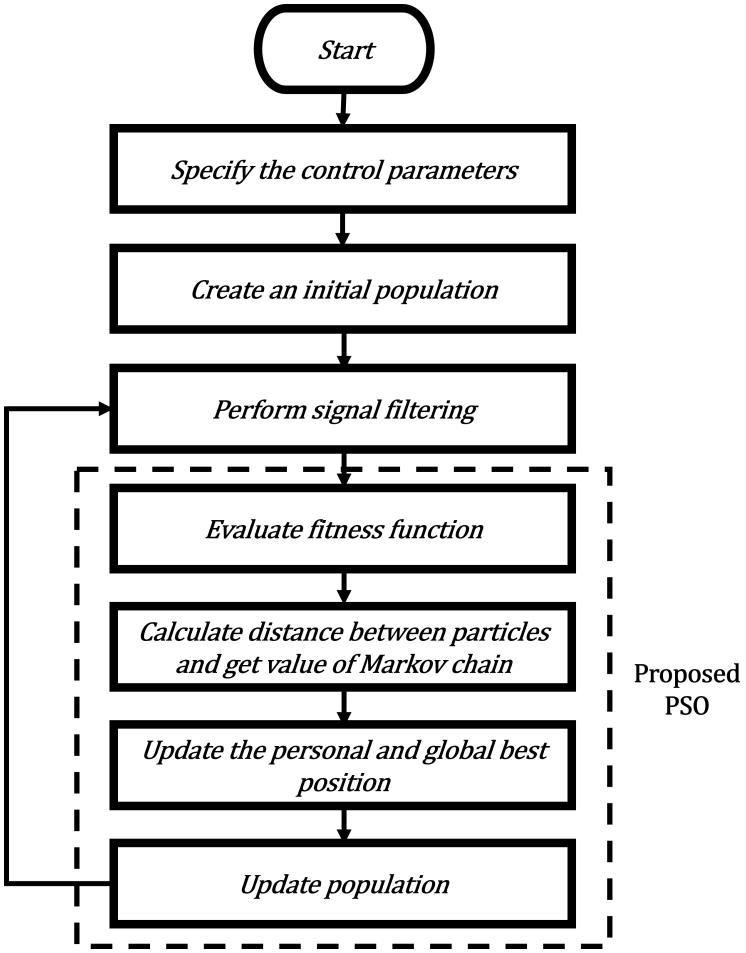

Figure 2. Flowchart of the proposed Markov switching PSO algorithm.

To perform the proposed PSO algorithm, the following steps are required:

- *Specification of the control parameters*. Here, the proposed Markov switching PSO algorithm has a population matrix $\mathbf{W}$ with P adaptive filters, where each particle denotes an adaptive filter, as shown in Equation (1). Here, the order N of each adaptive filter determines the dimension of each particle. Therefore, the whole population is defined as follows:

$$\mathbf{W} = \begin{bmatrix} w_1^1 & w_1^2 & \dots & w_1^P \\ w_2^1 & w_2^2 & \dots & w_2^P \\ \vdots & \vdots & \ddots & \vdots \\ w_N^1 & w_N^2 & \dots & w_N^P \end{bmatrix} \tag{1}$$

- *Creation of the initial population*. At the first iteration $n = 1$, the position $\mathbf{w}_i(n)$ of each particle is initialized, where $i = 1, 2, \dots, P$.

$$\mathbf{w}_i(n) = (ub - lb) \cdot \mathbf{r} + lb \tag{2}$$

where $\mathbf{r}$ denotes a Gaussian process of length, N. The value lb is a lower bound and ub is an upper bound.

- *Calculation of the signal filtering* . The calculation of signal counteraction or also called residual noise $\mathbf{e}(n)$ is given by

$$\mathbf{e}_i(n) = \mathbf{d}(n) + \mathbf{y}_i(n) \tag{3}$$

$\mathbf{d}(\mathbf{n})$ denotes the desired signal and $\mathbf{y}(\mathbf{n})$ the filter output of the i-th filter.

- *Evaluation of the fitness function*. To compute the best position, the PSO algorithm uses the mean squared error (MSE) of each P error signal as a fitness function of each adaptive filter. The evaluation of the position $\mathbf{w}_i(n)$ can be computed as follows:

$$f_i(n) = \frac{1}{N}\sum_{k=1}^{N} e_i^2(k) \tag{4}$$

- *Calculation of the distance between particles and the obtention of the value of Markov chain*. Here, the velocity and position are obtained by using the following equations:

$$\mathbf{v}_i(n) = \phi \cdot \mathbf{v}_i(n-1) + c_1(\xi(n)) \cdot r_1[\mathbf{w}_{pbest_i} - \mathbf{w}_i(n)] + c_2(\xi(n)) \cdot r_2[\mathbf{w}_{gbest} - \mathbf{w}_i(n)] \tag{5}$$

$$\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + \mathbf{v}_i(n) \tag{6}$$

where r_1 and r_2 are the vectors of random numbers of length N, $\mathbf{w}_{pbest_i}$ and $\mathbf{w}_{gbest}$ are the personal best position and global best position, respectively, and ϕ the inertia weight. Here, we define r_1 and r_2 in the interval $[0, 1]$. $c_1(\xi(n))$ and $c_2(\xi(n))$ are the acceleration coefficients determined by a non-homogeneous Markov chain $\xi(n)(n \geq 0)$. The value of the Markov chain is taken in a finite state space: $S = \{1, 2, \ldots, L\}$. $\Pi(n) = (\pi_{ij}(n))_{LxL}$, where $\Pi(n)$ denotes the probability transition matrix of the Markov chain, where $\pi_{ij}(n) \geq 0 (i, j \in S)$ and $\sum_{j=1}^{N} \pi_{ij}(n) = 1$. It is important to keep in mind that the matrix Π is dynamically adjusted by evaluating an evolutionary factor (E_f) [14] according to the population distribution properties [15]. Based on these characteristics, the E_f approach can be exploited at the maximum to define four states: convergence, exploration, exploitation and jumping out. In particular, these four states are, respectively, represented by $\xi(n) = 1$, $\xi(n) = 2$, $\xi(n) = 3$ and $\xi(n) = 4$ in the Markov chain.
The average distance, d_i, between each particle and the other particles is computed as follows:

$$d_i(n) = \frac{1}{P}\sum_{j=1}^{P} \sqrt{\sum_{k=1}^{N} (x_i(k) - x_j(k))^2} \tag{7}$$

where P and N denotes the swarm size and the dimensions of each particle, respectively. Hence, the evolutionary factor E_f can be obtained as follows [15]:

$$E_f = \frac{d_g - d_{min}}{d_{max} - d_{min}} \tag{8}$$

where d_g represents the globally best particle among d_i. d_{max} and d_{min} are the maximum and minimum distances in d_i, respectively.
Here, we obtain the value of the Markov chain, which is based on the value of evolutionary factor E_f, as follows [14]:

$$\xi(n) = \begin{cases} 1, & 0 \leq E_f < 0.25, \\ 2, & 0 \leq E_f < 0.5, \\ 3, & 0 \leq E_f < 0.75, \\ 4, & 0 \leq E_f < 1, \end{cases} \tag{9}$$

where the probability transition matrix is given by:

$$\Pi = \begin{pmatrix} \chi & 1-\chi & 0 & 0 \\ \frac{1-\chi}{2} & \chi & \frac{1-\chi}{2} & 0 \\ 0 & \frac{1-\chi}{2} & \chi & \frac{1-\chi}{2} \\ 0 & 0 & 1-\chi & \chi \end{pmatrix} \tag{10}$$

Based on the probability distribution matrix Π, the Markov process may switch its state at the next iteration. To guarantee the classification accuracy and the search diversity, the value of the probability χ is equal to 0.9 [14]. Here, the initial values of acceleration coefficients c_1 and c_2 are selected by trial-and-error for all states in order to guarantee the best performance of the purposed algorithm. Table 1 shows their values based on the evolutionary state, which are automatically adjusted.

Table 1. Strategies for selecting c_1 and c_2.

State	Mode	c_1	c_2
Convergence	$\xi(n) = 1$	2	2
Exploitation	$\xi(n) = 2$	2.1	1.9
Exploration	$\xi(n) = 3$	2.2	1.8
Jumping-out	$\xi(n) = 4$	1.8	2.2

- *Update the personal and global best position*. To get the value of the personal best $\mathbf{w}_{pbest_i}(n)$, a comparison between the current value of $f_i[\mathbf{w}_i(n)]$ and the value of $f[\mathbf{w}_{pbest_i}(n-1)]$ is performed as follows:

$$\mathbf{w}_{pbest_i}(n) = \begin{cases} \mathbf{w}_i(n), & \text{if } f_i[\mathbf{w}_i(n)] < f[\mathbf{w}_{pbest_i}(n-1)] \\ \mathbf{w}_{pbest_i}(n-1), & \text{otherwise} \end{cases} \tag{11}$$

The $\mathbf{w}_i(1)$ defined as $\mathbf{w}_{pbest_i}(1)$ is used to calculate Equation (11) at the first generation. To calculate the global best position, $\mathbf{w}_{gbest}$, we compare the result of $f[\mathbf{w}_{pbest_{min}}(n)]$ with the evaluation of the best global position $f[\mathbf{w}_{gbest}(n-1)]$, where $\mathbf{w}_{pbest_{min}}(n) = \mathbf{w}_{pbest_g}(n)$, $g = arg\ min_{1 \le i \le P}\{\mathbf{w}_{pbest_{j,i}}(n)\}$. The computation of the global best position, $\mathbf{w}_{gbest}$ is obtained as follows:

$$\mathbf{w}_{gbest}(n) = \begin{cases} \mathbf{w}_{pbest_{min}}(n), & \text{if } f[\mathbf{w}_{pbest_{min}}(n)] < f[\mathbf{w}_{gbest}(n-1)] \\ \mathbf{w}_{gbest}(n-1), & \text{otherwise} \end{cases} \tag{12}$$

- *Update population*. Equations (5) and (6) are used to update the velocity and position of each particle, respectively, and Equation (13), which is in the function of the power of the instantaneous error, is used to update the population size.

$$P = \lfloor \frac{2(P_{max} - P_{min})}{1 + e^{-e(n)^2}} - (P_{max} - P_{min}) \rfloor + P_{min} \tag{13}$$

where P_{max} and P_{min} are the maximum and minimum number of particles, respectively.

3. Pure Software Implementation

Before implementing the proposed Markov switching PSO algorithm in parallel hardware architectures, we simulate it in Matlab software for testing and comparison purposes. Here, we use the AEC structure, in which the existing approaches and the proposed adaptive filter are 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 Markov switching PSO algorithm, we consider the following conditions:

- We use an impulse response as an echo path obtained from the ITU-T G168 recommendation [16]. This echo path is modeled using N = 500 coefficients, as shown in Figure 4.
- The echo signal is mixed with white Gaussian noise (SNR = 20 dB).
- The input signal is an AR(1) process, which is produced by filtering white Gaussian noise by means of the system $\frac{1}{(1-0.95z^{-1})}$.

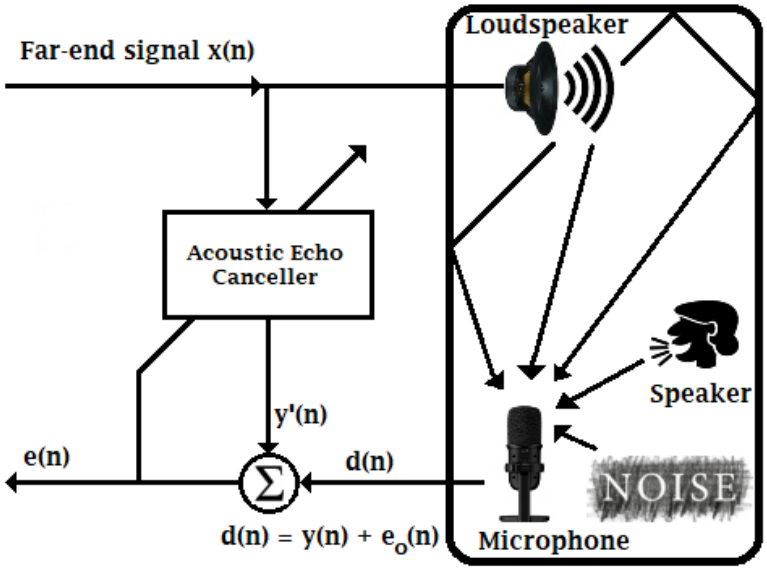

Figure 3. Structure of Acoustic Echo Canceller.

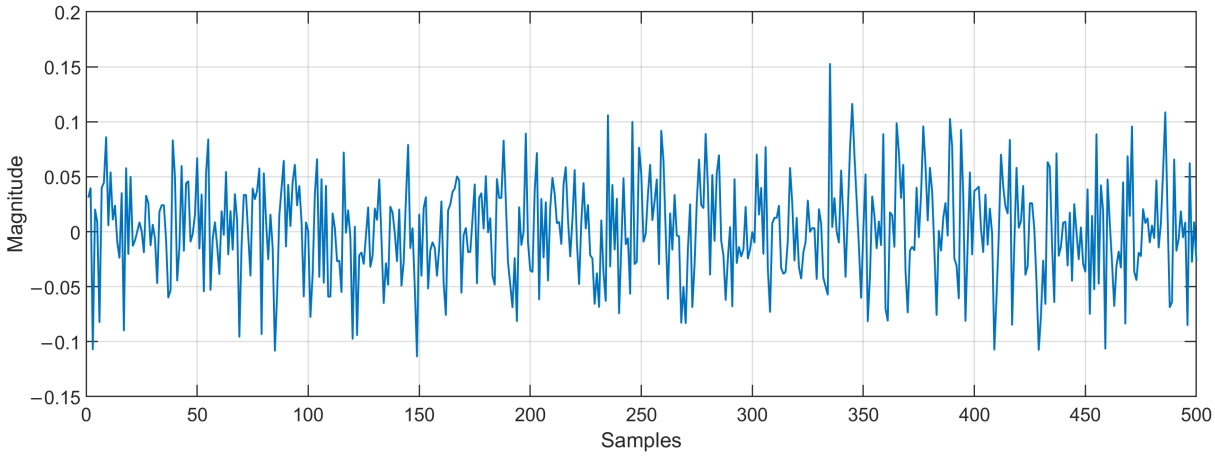

Figure 4. Impulse response of the acoustic echo path.

- In the proposed Markov switching PSO algorithm, the swarm size is defined in the range of 100–20 particles, while the swarm size, which is used in the simulation of an existing approach, is set to 100.
- To probe the tracking capabilities of the proposed Markov switching PSO algorithm, 4 different experiments were simulated: (1) Changing SNR from 20 dB to 10 dB in the middle of iterations, (2) causing an abrupt change to the impulse response of the acoustic echo path in the middle of the adaptive filtering process by multiplying the acoustic path by -1, (3) causing an abrupt change to the impulse response of the acoustic echo path in the middle of the adaptive filtering process by shifting the acoustic path, and (4) simulating a double talk-scenario at the middle of iterations.
- Acceleration coefficients of the conventional PSO were selected to obtain the best performance.
- The maximum number of iterations is set to 4,000,000.
- We verify the performance of the proposed algorithm in terms of echo return loss enhancement, ($ERLE = 10log_{10}(\frac{d(n)^2}{e(n)^2})$).

Here, we simulate the conventional PSO [17] and the proposed Markov switching PSO algorithm to compare their performance. As can be observed from Figure 5, the proposed

algorithm shows better performance in comparison with the conventional PSO in terms of echo return loss enhancement (ERLE) and convergence speed.

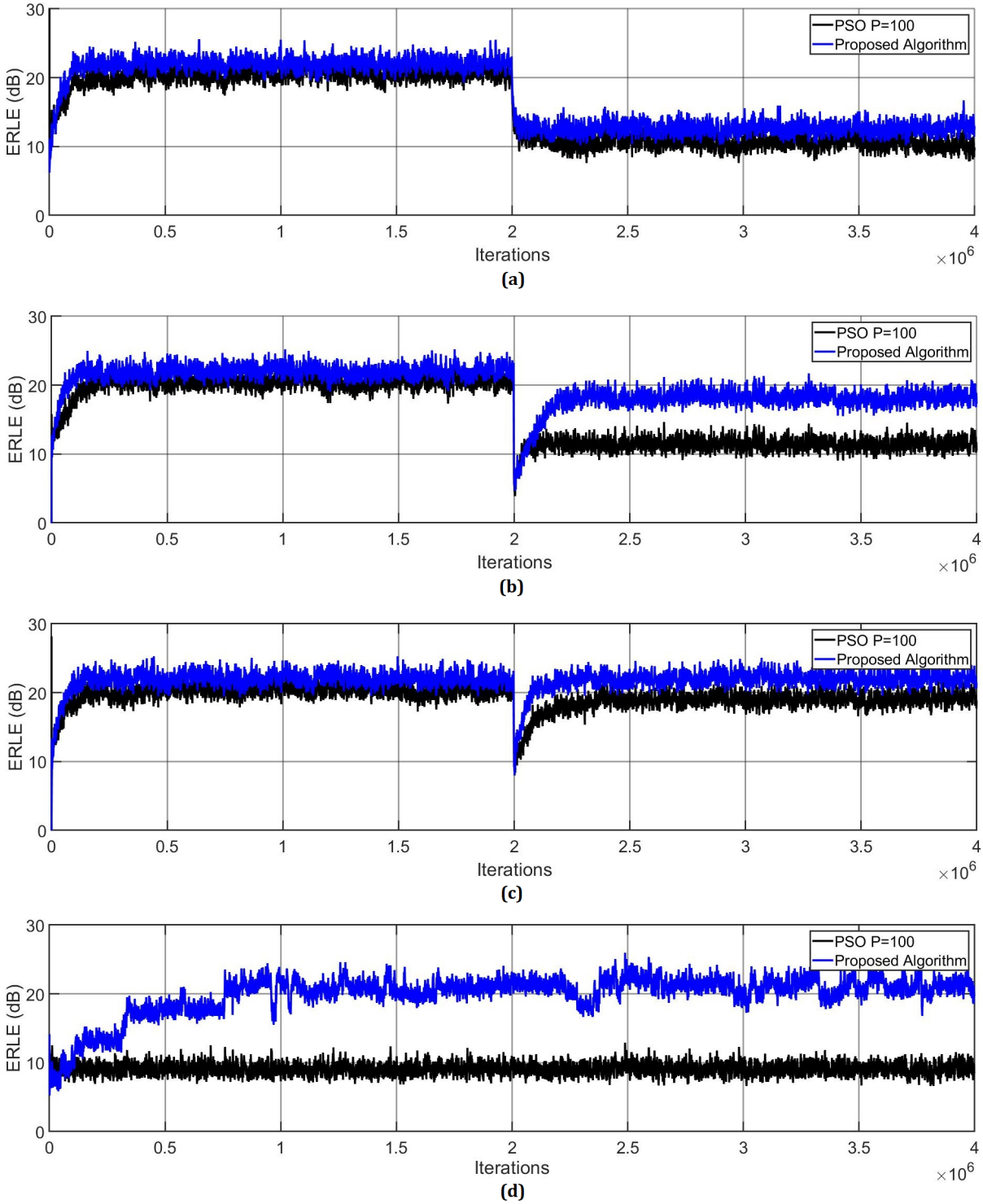

Figure 5. ERLE of the proposed Markov switching PSO algorithm and conventional PSO algorithms by computing the AR(1) process input signal. (**a**) Changing the SNR from 20 dB to 10 dB at the middle of the iterations. (**b**) Multiplying the acoustic path by -1 in the middle of the adaptive filtering process. (**c**) Shifting the acoustic path in the middle of the adaptive filtering process. (**d**) Simulating the double-talk scenario.

To make a coherent comparison between the proposed algorithm and existing algorithms we carried out an experiment in which the acoustic path is multiplied by -1 midway through the iterations. The algorithms that were simulated for this comparison and their tuning parameters are described in the following list:

1. Grey wolf optimization (GWO) [18]
   - Population size $= 50$
   - lower bound $= -1$
   - Upper bound $= 1$
   - a decreases linearly from 2 to 0
2. PSO [19]
   - Population size $= 100$
   - Lower bound $= -1$
   - Upper bound $= 1$
   - Acceleration coefficient, $c_1 = 1.6$
   - Acceleration coefficient, $c_2 = 1$
   - Inertia weight $= 0.8$
3. Differential evolution (DE) [20]
   - Population size $= 50$
   - Lower bound $= -1$
   - Upper bound $= 1$
   - Crossover rate $= 0.35$
   - Scaling factor $= 0.8$
   - Combination factor $= 0.25$
4. Artificial bee colony optimization (ABC) [21]
   - Population size $= 50$
   - Lower bound $= -1$
   - Upper bound $= 1$
   - Evaporation parameter $= 0.1$
   - Pheromone $= 0.6$
5. Hybrid PSO–LMS [22]
   - Population size $= 60$
   - Lower bound $= -1$
   - Upper bound $= 1$
   - Acceleration coefficient, $c_1 = 0.00005$
   - Acceleration coefficient, $c_2 = 1.2$
   - Inertia weight $= 1$
   - Convergence factor $= 1 \times 10^{-9}$
6. Modified ABC (MABC) [23]
   - Population size $= 50$
   - Lower bound $= -1$
   - Upper bound $= 1$
   - Evaporation parameter $= 0.1$
   - Pheromone $= 0.6$
   - Convergence factor $= 3 \times 10^{-5}$

As can be observed from Figure 6, the proposed algorithm shows the best performance in terms of convergence speed and ERLE level. In addition, the proposed algorithm requires a lower computational cost when compared with the GWO, PSO, ABC, PSO-LMS and MABC algorithms, as shown in Table 2. To obtain these data, we consider a double-talk scenario, in which the proposed Markov switching PSO algorithm requires fewer multiplications and additions when compared with most of the existing algorithms during the whole simulation. The reason for this is that the number of particles, P, which are used

to model the proposed Markov switching PSO algorithm, is reduced during the filtering process. Specifically, we initially use 100 particles, after some iterations, this number is reduced to 20 particles. In contrast, the existing approaches need a larger population to obtain acceptable performance at the cost of increasing the computational burden. In AEC applications, this aspect is crucial because one of the objectives of the proposed Markov switching PSO algorithm is to improve the computational cost without losing performance and this is possible by using the Markov Switching technique.

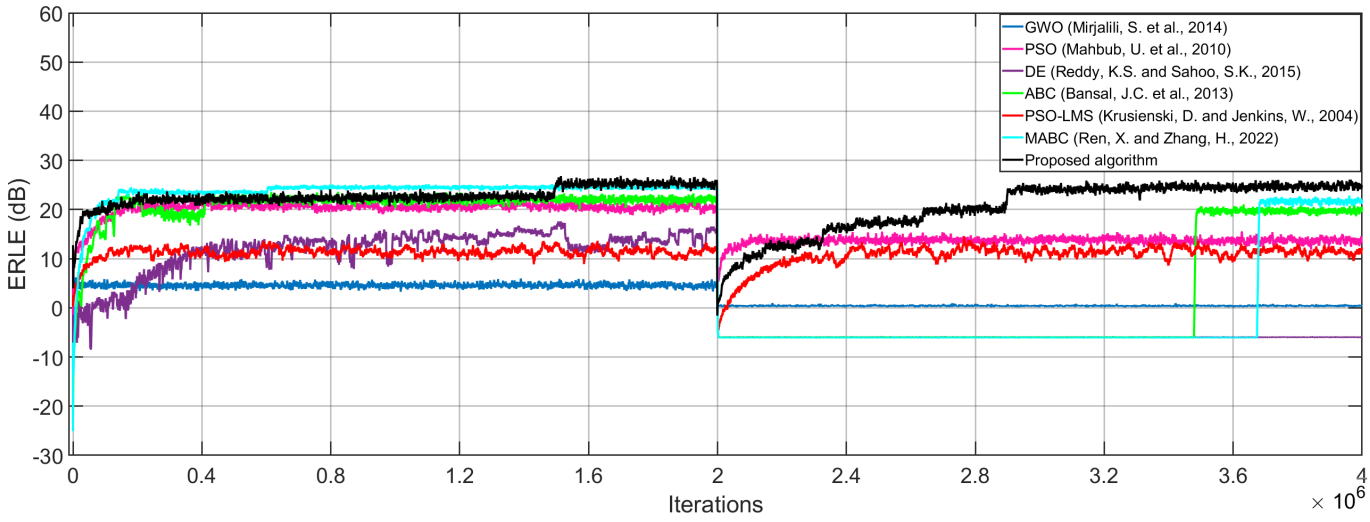

Figure 6. ERLE of the proposed Markov switching PSO algorithm and existing approaches [18–23] by computing the AR(1) process input signal when multiplying the acoustic path by -1 at the middle of the adaptive filtering process.

Table 2. Number of additions and multiplications required by the conventional PSO algorithm [17] and the proposed algorithm.

Operation	Algorithm	Equation	Number of Operations
Addition	GWO	$9NP + 1$	4.5000×10^{14}
	PSO	$5NP + 2$	5.0000×10^{14}
	DE	$2NP$	1.0000×10^{14}
	ABC	$5NP - 3$	2.4999×10^{14}
	PSO-LMS	$5NP + 2N + 2$	3.0200×10^{14}
	MABC	$5NP + 2N - 2$	3.0200×10^{14}
	Proposed Algorithm	$7NP + 8$	2.2271×10^{14}
Multiplication	GWO	$15NP + 1$	7.5000×10^{14}
	PSO	$5NP + 2$	5.0000×10^{14}
	DE	NP	5.0000×10^{13}
	ABC	$4NP + 3$	2.0001×10^{14}
	PSO-LMS	$5NP + 4N$	3.0101×10^{14}
	MABC	$4NP + 2N + 4$	2.4201×10^{14}
	Proposed Algorithm	$7NP + 7$	2.2271×10^{14}

4. Hardware Implementation

Once the performance of the proposed Markov-PSO adaptive filter was verified, we develop a parallel metaheuristic processor to simulate it at high processing speeds by expanding the minimum area consumption. To achieve a low area consumption; first, we use the time multiplexing technique to simulate particles virtually; second, we use optimized neural multipliers and adders since is the most demanding arithmetic circuit in terms of area and processing speed. In addition, the use of these circuits has allowed us to optimize the processing time since each operation is performed at a single clock cycle [24]. As can be observed from Figure 7, the proposed parallel metaheuristic processor is mainly composed of the following components:

- *Markov-PSO processing core, M-PSO PC*. This represents the basic processing element to compute the signal-filtering process and update the population. The proposed M-PSO PC mainly uses neural multipliers Π_{mul} [24] and adders Π_{add} [24]. Additionally, this circuit has a slave control unit CU_{s1}, pseudo-random number generators, RNG, and a Markov processor core, MP (Figure 8). In particular, the MP core is in charge of performing the calculation of the distance between particles by means of the optimized square root circuit [25], as shown in Figure 9.
- *A master control unit, CU_m*. This module is in charge of controlling the data flow and synchronization. Specifically, this component performs the time multiplexing technique to simulate several particles at different times by using the same *M-PSO PC*. In addition, this component sends the control signals to store the input samples, $x(n)$, in the BRAMs. In this way, the block-processing technique can be properly implemented.
- *Distribution module, DM*. The main function of this component is to evaluate and indicate the personal and global best. Therefore, this component transfers this information to each *M-PSO PC* in parallel. Therefore, the update process can be done at high processing speeds.

It should be noted that we presented a neural adder circuit, $\prod_{add}$, and a neural multiplier, $\prod_{mul}$, to perform addition and multiplication of fixed-point numbers, respectively, in [24]. To implement the AEC system in an FPGA device, the fixed point representation is highly demanded since the simulation of metaheuristic algorithms requires high-precision calculations. Therefore, we created advanced neurons, which are based on spiking neural P systems, by improving their structural and functional capabilities. In particular, we used cutting-edge variants of the SN P systems, such as anti-spikes, dendritic trunk, dendritic delays and rules on the synapses. As a result, we create high-precision neural adders and multipliers by employing a low number of synapses and neurons with simple spiking rules. In general, both circuits exhibit the following features:

- *Scalability*. These circuits can process numbers with any required length by only adding neurons in a regular and homogenous neural structure.
- *Compactness*. To obtain a great improvement in terms of area, we designed the circuit by using a low number of neurons and synapses. Specially, we optimized the number of synapses since the routing of a large number of synaptic connections creates place and routing problems, especially when they are implemented in advanced FPGAs.
- *High performance*. In this application, the real-time filtering process is highly demanded. Therefore, we achieved neural multiplier and adder to perform their respective operations by expending a single and ten clock cycles, respectively.

Once the proposed parallel metaheuristic processor was debugged, we integrate it into the structure of the AEC system to validate its performance, as shown in Figure 10.

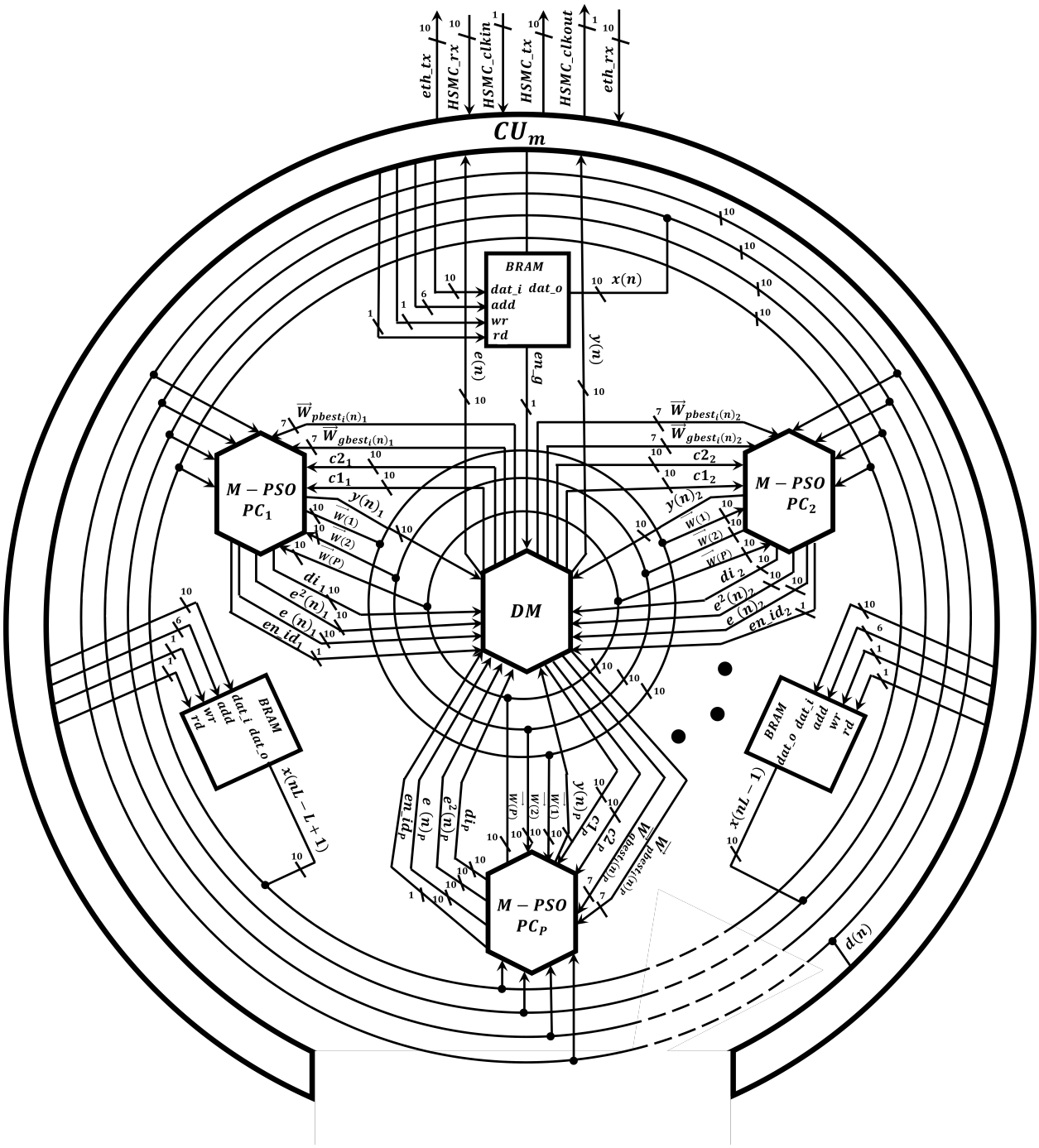

Figure 7. Digital implementation of the proposed Markov switching PSO algorithm in the parallel metaheuristic processor.

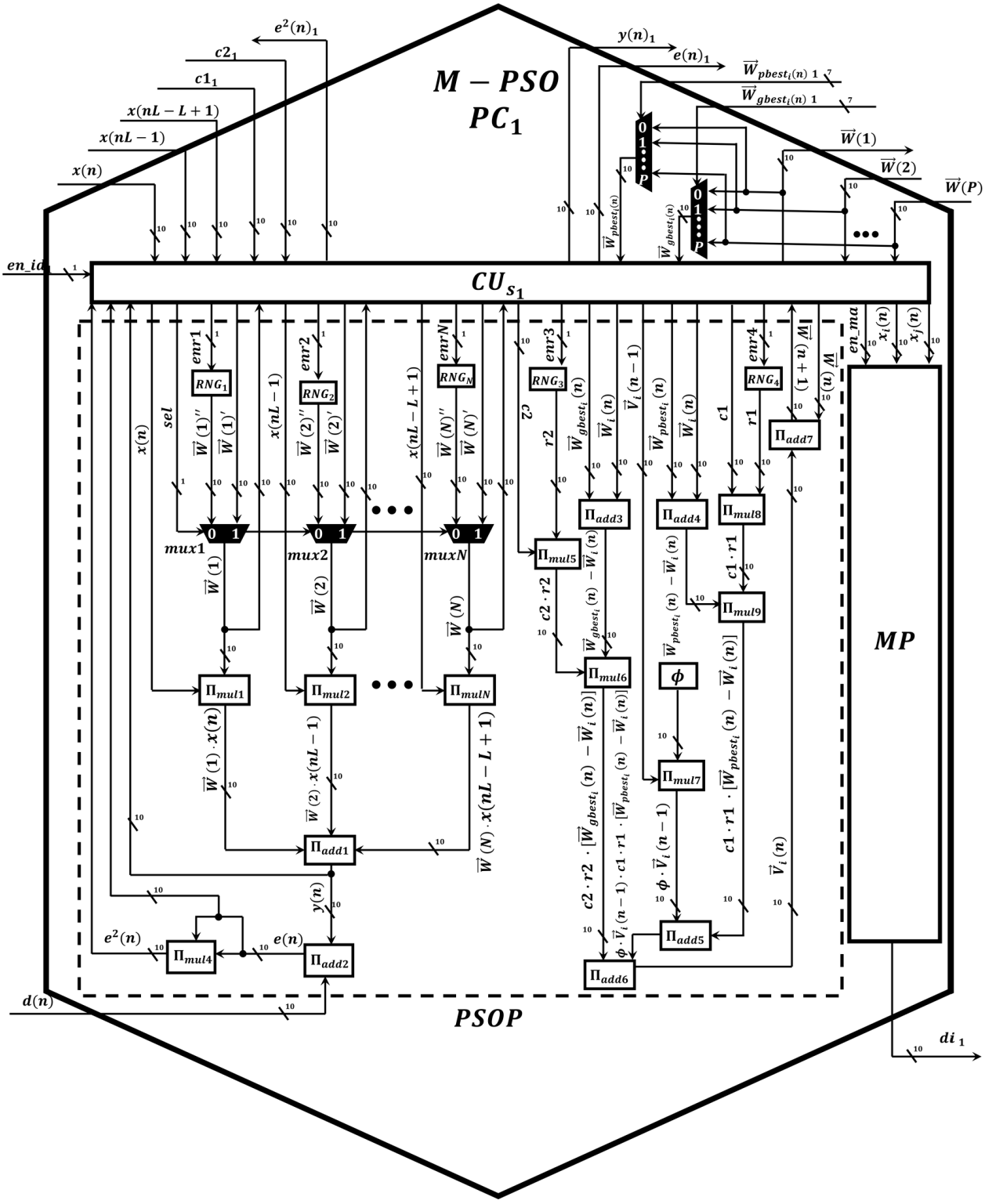

Figure 8. Digital circuit scheme of the Markov-PSO processing core.

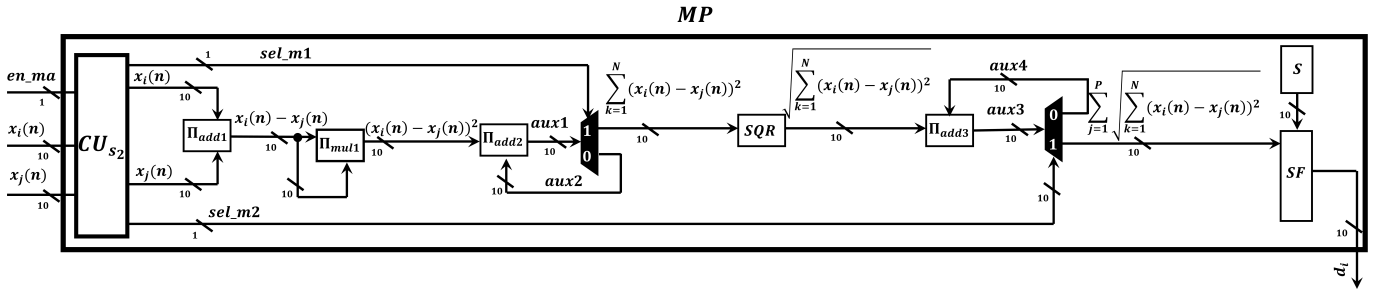

Figure 9. Digital circuit scheme of the Markov proccesor.

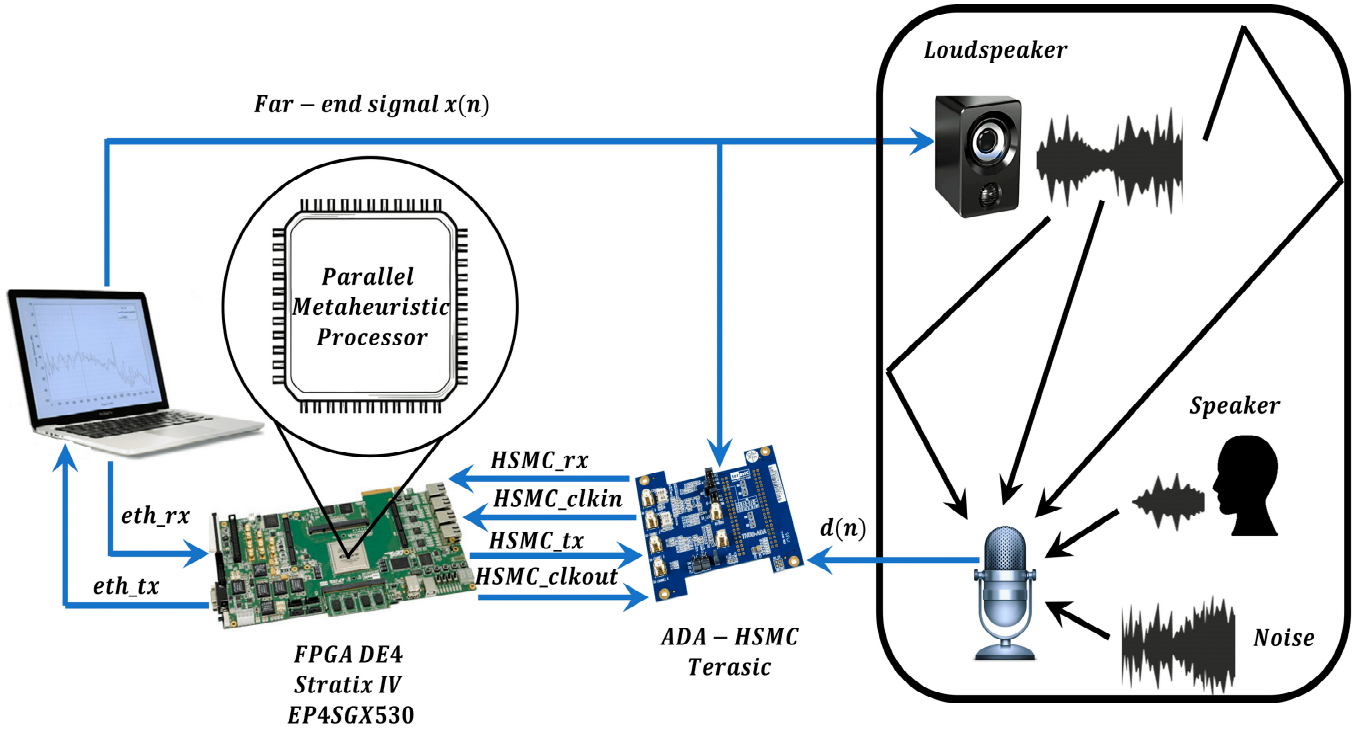

Figure 10. General scheme of the components of the AEC system.

To demonstrate the computational capabilities of the meta-heuristic processor, we develop two sets of experiments. Particularly, we simulate single-talk and double-talk scenarios by employing two different input signals, $x(n)$. In addition, under these scenarios, the proposed metaheuristic processor computes the AR(1) process and the speech signal by considering an under-modeling case. Specifically, we employ 512 coefficients to model the adaptive filter, while the echo path [26] is configured by using 1024 coefficients. In real-world echo noise applications, the performance of the AEC can be crucially decreased by the variations of background noise. For a single-talk scenario, we decreased the SNR from 20 to 10 dB in the middle of the iterations, as shown in Figure 11. It should be noted that the background noise variation does not affect the performance of the proposed algorithm. On the other hand, the proposed metaheuristic processor was implemented in Stratix IV GX EP4SGX530 FPGA. Here, this implementation, which involves the use of eight BRAMs and 20 *M-PSO PCs*, requires 384,748 LEs. This represents 72.429% of the total area of the FPGA. In this way, we can simulate 100 particles virtually since each *M-PSO PC* simulates five particles serially. The processing time to simulate all of these particles is 89.1 µs, which are obtained by multiplying the number of clock cycles (11,143), which are obtained by means of Equation (14), by the system clock period (8 ns). It should be noted that the required processing time in the FPGA device is less in comparison when this algorithm is simulated in a server, which includes a Xeon E5-2630 processor working at 2.6 GHz

and 64 Gb RAM, since the simulation of the algorithm on this computer requires 1.47 ms, considering the simulation of 100 particles. This can be considered the worse case since the number of particles decreases over the filtering process. As a consequence, the processing time also decreases. This factor is vital in the simulation of real-time AEC systems since the maximum latency of the system is 125 µs, i.e., the input signal is sampled at 8 KHz. On the other hand, the simulation of the proposed Markov-PSO adaptive filter consumes up to 328 mW, by considering the worst case (one hundred particles). After observing the results of the above experiments, we prove that the metaheuristic processor is capable of processing a variable number of particles to perform the proposed Markov switching PSO algorithm at high processing speeds.

$$N_{cc} = 128 + (y - 12 + 1 + x) \cdot 5 + (y - 12 + 1 + x) \cdot \frac{x}{20} + (1000 + 1) \cdot \frac{x}{20} \quad (14)$$

where y represents the number of coefficients and x depicts the number of particles.

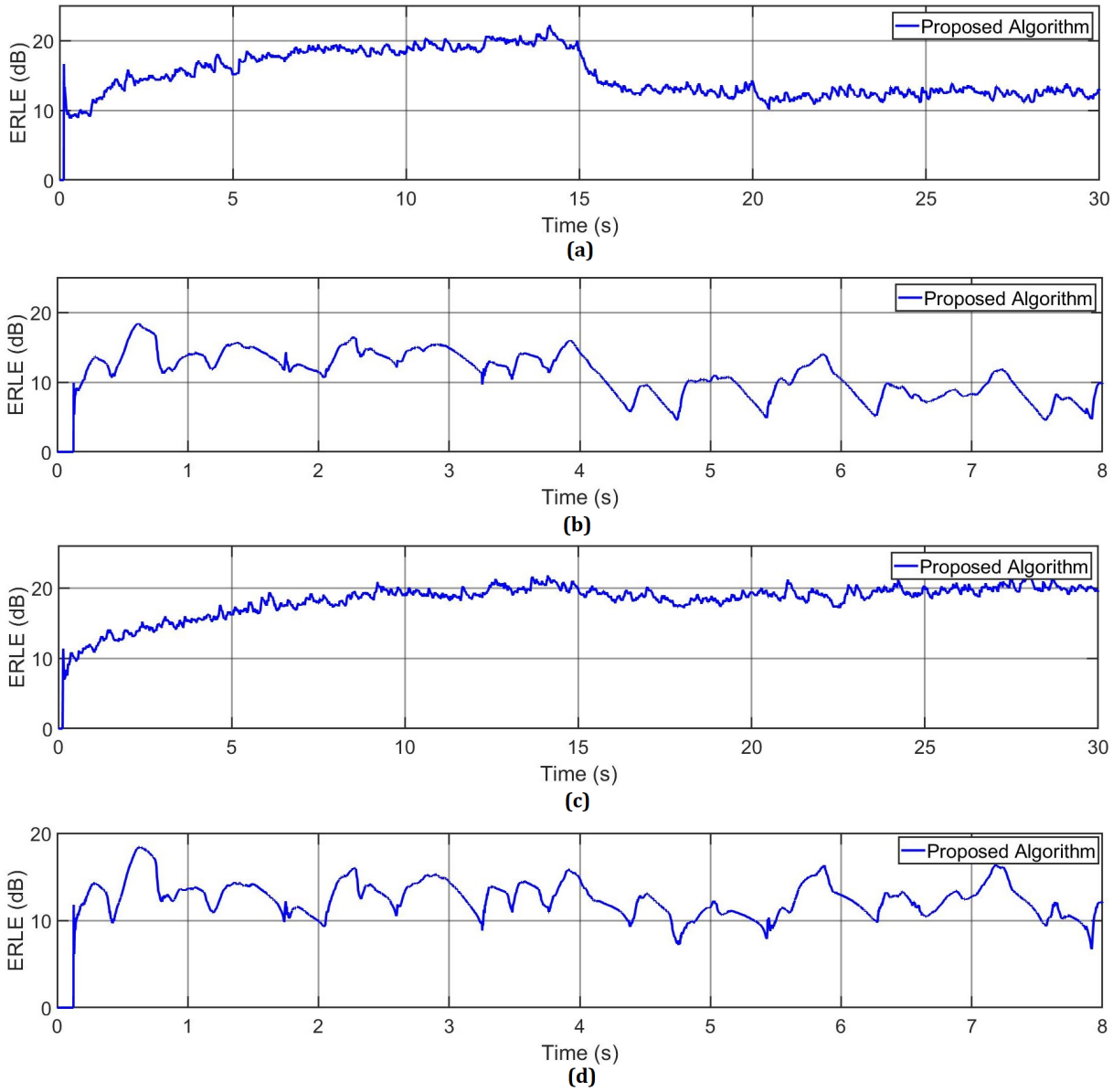

Figure 11. (**a**) AR(1) process and (**b**) speech signal used in a single-talk scenario; (**c**) AR(1) process and (**d**) speech signal used in a double-talk scenario.

5. Conclusions

In this work, we present, for the first time the development of a high-speed and compact FPGA-based parallel metaheuristic process to efficiently simulate a new variant of the PSO algorithm based on the Markovian switching technique. Here, we grouped our contributions as follows:

- *From the AEC model point of view*. In this work, we made intensive efforts to reduce the computational cost of the AEC systems to be implemented in resource-constrained devices. In addition, we significantly improve the convergence properties of these systems by using an improved metaheuristic swarm intelligence method to be used in practical acoustic environments. Specifically, we present a new variant of the PSO algorithm based on the Markovian switching technique. The use of this technique has allowed us to guarantee a higher convergence rate and higher ERLE in comparison when the conventional PSO algorithm is used. To make feasible the implementation of the proposed variant of the PSO algorithm in embedded devices, we use the block-processing scheme. In this way, the proposed algorithm can be easily implemented in parallel hardware architectures. As a consequence, it can be simulated at high processing speeds. In addition, we significantly reduce the computational cost of the proposed conventional PSO algorithm. To achieve this aim, we propose a method to dynamically decrease the number of particles of this new variant of the PSO algorithm over the filtering process.
- *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 particles by using the proposed time-multiplexing control scheme. In this way, we properly implement the proposed Markov switching PSO algorithm, in which the number of particles decreases according to the simulation needs, in a Stratix IV GX EP4SGX530 FPGA.

Finally, we carry out several experiments to prove that the proposed Markov switching PSO algorithm along with new techniques potentially allows the creation of practical and real-time AEC processing tools.

Author Contributions: Conceptualization, G.S. (Giovanny Sánchez) and J.C.S.; Formal Analysis, Methodology and Supervision, J.G.A. and G.S. (Giovanny Sánchez); Software, Validation, Investigation, E.A., G.S. (Guillermo Salinas) and E.P.; Writing-Reviewing and Editing, G.S. (Guillermo Salinas) and E.P.; Project Administration, J.C.S. and E.V.; Resources, Funding Acquisition, G.S. (Gabriel Sánchez) and L.K.T. 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 carry out this work.

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

References

1. 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.
2. 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 IEEE International Conference on Electrical & Computer Engineering (ICECE 2010), Dhaka, Bangladesh, 18–20 December 2010; pp. 518–521.
3. 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. [CrossRef]

4. Kimoto, M.; Asami, T. Multichannel Acoustic Echo Canceler Based on Particle Swarm Optimization. *Electron. Commun. Jpn.* **2016**, *99*, 31–40. [CrossRef]
5. Tang, J.; Zhao, X. Particle swarm optimization with adaptive mutation. In Proceedings of the 2009 IEEE WASE International Conference on Information Engineering, Taiyuan, China, 10–11 July 2009; Volume 2, pp. 234–237.
6. Ratanavilisagul, C.; Kruatrachue, B. Selective crossover base on fitness in multiswarm optimization. In Proceedings of the International Conference on Emerging Trends in Computer and Image Processing (CIP'11), Bangkok, Thailand, 11–14 September 2011; pp. 12–15.
7. Chi, Y.; Cai, G. Particle swarm optimization with opposition-based disturbance. In Proceedings of the 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010), Wuhan, China, 6–7 March 2010; Volume 2, pp. 223–226.
8. Mauryan, R.; Thanushkodi, K.; Sakthisuganya, A. Reactive power optimization using quantum particle swarm optimization. *J. Comput. Sci.* **2012**, *8*, 1644–1648.
9. Lu, S.; Yu, S. An improved particle swarm optimizer with attraction and repulsion. In Proceedings of the 2012 7th International Conference on Computing and Convergence Technology (ICCCT), Seoul, Republic of Korea, 3–5 December 2012; pp. 735–740.
10. Arani, B.O.; Mirzabeygi, P.; Panahi, M.S. An improved PSO algorithm with a territorial diversity-preserving scheme and enhanced exploration–exploitation balance. *Swarm Evol. Comput.* **2013**, *11*, 1–15. [CrossRef]
11. Zermani, A.; Manita, G.; Feki, E.; Mami, A. Hardware implementation of particle swarm optimization with chaotic fractional-order. *Neural Comput. Appl.* **2023**, 1–20. [CrossRef]
12. Da Costa, A.L.; Silva, C.A.; Torquato, M.F.; Fernandes, M.A. Parallel implementation of particle swarm optimization on FPGA. *IEEE Trans. Circuits Syst. II Express Briefs* **2019**, *66*, 1875–1879. [CrossRef]
13. Shaikh, U.T.; Kalwar, I.H.; Memon, T.D.; Shaikh, F. Design of IIR filter using PSO algorithm and its implementation in FPGA. *Indian J. Sci. Technol.* **2017**, *10*, 1–5. [CrossRef]
14. Tang, Y.; Wang, Z.; Fang, J.A. Parameters identification of unknown delayed genetic regulatory networks by a switching particle swarm optimization algorithm. *Expert Syst. Appl.* **2011**, *38*, 2523–2535. [CrossRef]
15. Zhan, Z.H.; Zhang, J.; Li, Y.; Chung, H.S.H. Adaptive particle swarm optimization. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2009**, *39*, 1362–1381. [CrossRef] [PubMed]
16. lnternational Telecommunication Union ITU-T. Digital Network Echo Cancellers. In *Standardization Sector of ITU*; lnternational Telecommunication Union ITU-T: Geneva, Switzerland, 2002.
17. Rout, N.K.; Das, D.P.; Panda, G. Particle swarm optimization based active noise control algorithm without secondary path identification. *IEEE Trans. Instrum. Meas.* **2011**, *61*, 554–563. [CrossRef]
18. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
19. Krusienski, D.; Jenkins, W. Adaptive filtering via particle swarm optimization. In Proceedings of the The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, Pacific Grove, CA, USA, 9–12 November 2003; pp. 571–575.
20. 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]
21. Bansal, J.C.; Sharma, H.; Jadon, S.S. Artificial bee colony algorithm: A survey. *Int. J. Adv. Intell. Paradig.* **2013**, *5*, 123–159. [CrossRef]
22. 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.
23. 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]
24. 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]
25. Hasnat, A.; Bhattacharyya, T.; Dey, A.; Halder, S.; Bhattacharjee, D. A fast FPGA based architecture for computation of square root and Inverse Square Root. In Proceedings of the 2017 Devices for Integrated Circuit (DevIC), Kalyani, India, 23–24 March 2017; pp. 383–387.
26. Paleologu, C.; Ciochina, S.; Benesty, J. Variable step-size NLMS algorithm for under-modeling acoustic echo cancellation. *IEEE Signal Process. Lett.* **2008**, *15*, 5–8. [CrossRef]