IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 August 2023, accepted 17 September 2023, date of publication 22 September 2023,
date of current version 28 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3317896

RESEARCH ARTICLE

Parallelized Particle Filter With Efficient Pipelining on FPGA for Real-Time Ballistic Target Tracking

DAEYEON KIM[1], HEONCHOEL LEE[1], (Member, IEEE), HYUCK-HOON KWON[2], YEJI HWANG[2], AND WONSEOK CHOI[2]

[1]Department of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi-si 39177, South Korea
[2]PGM Research and Development Laboratory, LIG Nex1, Seongnam-si 13488, South Korea

Corresponding author: Heonchoel Lee (hclee@kumoh.ac.kr)

This work was supported by the Theater Defense Research Center funded by the Defense Acquisition Program Administration under Grant UD200043CD.

ABSTRACT In this paper, the problem of a real-time ballistic target tracking is addressed. Because the tracking process can be affected by the uncertainty in models and disturbances, probabilistic filters have been applied to solving the ballistic target tracking problem. Particle filters (PFs) can be used as a probabilistic filter because of their advantages of dealing with non-Gaussian models. However, since the PFs require too much computational costs, it is hard to apply the PFs to real-time systems for ballistic target tracking. The Graphic Processor Unit (GPU) can be considered to accelerate the PFs in the parallelization manner. But, the GPU-based parallelization requires too much power consumption, which means that it is difficult to be applied to the ballistic target tracking system due to the heat problem caused by the excessively increased power consumption. This paper proposes a practical approach which is a parallelized particle filter based on a heterogeneous processing system including a field-programmable gate array (FPGA) to accelerate the ballistic target tracking process by parallelizing the PFs while avoiding the heat problem. The evaluation results with a real heterogeneous processing system showed that the proposed approach could successfully conduct the ballistic target tracking and accelerate the PFs over four times.

INDEX TERMS Embedded systems, FPGA, parallel processing, particle filters, target tracking.

I. INTRODUCTION

Ballistic target tracking has been known as a crucial aspect of defense systems and has seen remarkable advancements with the advent of new methodologies. It involves the prediction and determination of the trajectory of a projectile or other object in flight, usually for the purposes of interception or evasion. One technique that has been prominently employed in this domain is the use of particle filters [1]. Several compelling advantages are offered by particle filters in solving the ballistic target tracking problem. Non-linear and non-Gaussian problems which are frequently encountered in real-world tracking scenarios [2] are handled adeptly by them. Unlike the Extended Kalman Filter (EKF) which assumes a Gaussian noise distribution and linear system dynamics, the particle filter is better suited for the complex dynamics and uncertainties found in ballistic target tracking [26]. Multi-modal distributions and multiple hypotheses, which permit more accurate and versatile tracking [3], are also features of particle filters. Moreover, their inherent recursiveness, which allows for real-time applications and continuous tracking without requiring all data to be present initially, is a key advantage [4].

However, despite these significant advantages, certain drawbacks are associated with the use of particle filters. The most notable among these is the computational intensity of particle filters, which can be prohibitive, particularly for real-time applications with resource-constrained devices [5]. This issue can be further compounded in scenarios where many particles are required to achieve acceptable estimation

The associate editor coordinating the review of this manuscript and approving it for publication was Remigiusz Wisniewski.

accuracy [6]. The challenges posed by these factors are addressed, and effective solutions to enhance the efficiency and accuracy of ballistic target tracking using particle filters are proposed in this paper. Most research has largely focused on the acceleration of Particle Filters using Graphics Processing Units (GPUs) [7], [8], [9]. Significant results in terms of increased computational speed have been yielded by this approach, but it causes the problem of power consumption [10]. The power consumption in GPU-based systems can be substantial, which can be particularly problematic when the target tracking system is integrated into an airborne platform.

Strict power constraints are often imposed on airborne platforms, establishing power efficiency as a critical concern alongside computational performance [28]. The high-power demand of GPU-based systems can, therefore, present a significant limitation in such applications. As a result, methodologies that both achieve computational acceleration and efficiently manage power consumption are needed.

Acceleration of the algorithm can primarily be achieved through three methods, each offering different approaches to enhance system performance. According to Table 1, the first method involves pipelining and parallelization, considering the spatio-temporal complexity of the algorithm. The implementation of the entire algorithm in hardware has been proposed in [11], [12], [13], and [14], while the pipelining and parallelization of resampling processes have been suggested in [15], [16], [17], and [18]. The second method focuses on algorithmic modifications, wherein the algorithm itself is optimized or transformed to enhance efficiency. Acceleration was achieved through a combination of pipelining, parallelization, and algorithmic modifications as described in [19], [20], [21], and [22]. Lastly, significant performance improvements can be realized by optimizing the interaction between hardware and software, thereby reducing associated overheads. In other words, efforts have been undertaken to reduce overheads, which have resulted in improvements. A Stream-like protocol for enhancement has been utilized in the proposed method.

Our main contributions are as follows:

1) A hardware-software co-design approach was proposed to accelerate the conventional particle filter for ballistic target tracking by parallelizing several parts, which can improve the real-time performance.
2) The unavoidable overhead resulting from data exchange between heterogeneous processors can be reduced by optimizing data exchange and communication protocols. Therefore, more efficient data transfer and processing between processors can be conducted.
3) The performance of the proposed approach was verified by successfully conducting co-simulations on a real evaluation board with heterogeneous processors.

The paper is organized to begin with a comprehensive description of the problem, where the application of the particle filter is introduced. This sets the stage for a thorough understanding of the context and the complexities involved. Next, the proposed method is explored in detail, with a focus on three techniques for hardware-software co-design. Each method is explained in detail. Finally, the evaluation results are presented, including a meticulous performance analysis and an in-depth ablation study. This section is intended to corroborate the effectiveness of the proposed methods and provide a tangible measure of their impact.

II. PROBLEM DESCRIPTION

The main goal of a tracking filter for a fast-moving object such as ballistic missiles is to estimate the target states in real time. At first, a ballistic missile trajectory is required in order to evaluate the performance of the tracking filter. Unlike outside the atmosphere, the aerodynamic forces such as drag in addition to gravity have an important influence on determining the ballistic missile trajectory in the reentry phase. We assume that the ballistic missile is a point mass describing in three-dimensional Cartesian coordinate system. A nonlinear dynamic equations including gravitational and aerodynamic forces as follows.

$$\dot{x} = V cos\gamma cos\psi \tag{1}$$
$$\dot{y} = V cos\gamma sin\psi \tag{2}$$
$$\dot{z} = -V sin\gamma \tag{3}$$
$$\dot{V} = \frac{(T - D - mgsin\gamma)}{m} \tag{4}$$
$$\dot{\gamma} = \frac{(Lcos\phi - mgcos\gamma)}{mV} \tag{5}$$
$$\dot{\psi} = \frac{Lsin\phi}{mVcos\gamma} \tag{6}$$

where $D = \frac{1}{2}\rho V^2 \cdot C_D \cdot S, L = \frac{1}{2}\rho V^2 \cdot C_L \cdot S$

As shown in Fig 1, x, y, z are position of missile, and V, γ, ψ represent velocity, flight path angle, heading angle, respectively. m is mass of missile, g is the gravitational acceleration, and T represent thrust. The aerodynamic forces D, L are drag and lift force, where consist of air density ρ, drag coefficient C_D, lift coefficient C_L, and reference area S. At last, ϕ means bank angle, which is the direction of lift force. In the paper, thrust is set to 0 and therefore mass is assumed to be constant because the combustion of propulsion system is generally completed before the reentry phase. In addition, since the effect of lift force is relatively small compared to drag force in non-maneuvering missile, L is assumed to be 0.

A. PARTICLE FILTER MODEL FOR REENTRY PHASE

For target-tracking estimations, several simplified target models have been proposed. In this paper, the widely used Singer model assuming that the target acceleration is a zero-mean, first-order, stationary Markov process was used [24] and [25]. The state-space formulation of the continuous-time Singer model is as follows.

$$\dot{\boldsymbol{x}} = F\boldsymbol{x} + G\boldsymbol{w} \tag{7}$$

TABLE 1. Overview of related works detailing various approaches to particle filter applications.

Reference	Method	Platform	Application	Target Device	Accelerated Part
[11]-[14]	Pipeline and parallelize	HW	Tracking	FPGA	Fully accelerated
[16]-[18]	Pipeline and parallelize	HW	Tracking	FPGA	Resampling
[7]	Pipeline and parallelize	HW/SW	Ballistic target tracking	CPU, GPU	Prediction
[15]	Pipeline and parallelize	HW/SW	Object tracking	CPU, FPGA	Resampling
[8]-[9]	Pipeline and parallelize	HW/SW	Object tracking	CPU, GPU	Resampling
[19]-[20]	Pipeline, parallelize and algorithmic modification	HW	Tracking	FPGA	Resampling
[21]	Pipeline, parallelize and algorithmic modification	HW	Tracking	FPGA	Fully accelerated
[22]	Pipeline, parallelize and algorithmic modification	HW/SW	Crack detection	CPU, FPGA	Fully accelerated
Proposed	Pipeline and parallelize	HW/SW	Ballistic target tracking	FPGA	Prediction

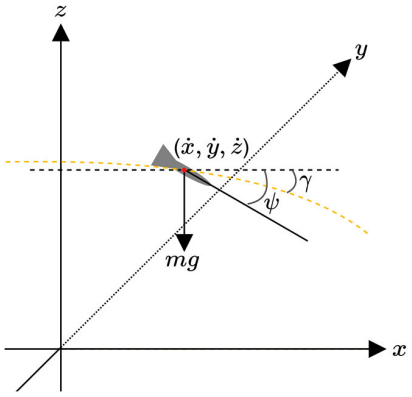


FIGURE 1. An illustration of the ballistic missile trajectory with parameters.

where

$$\boldsymbol{x} = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & a_x & a_y & a_z \end{bmatrix}^T$$

$$F = \begin{bmatrix} 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & I_3 \\ 0_3 & 0_3 & -I_3/\tau \end{bmatrix} \text{ and } G = \begin{bmatrix} 0_3 \\ 0_3 \\ I_3 \end{bmatrix}$$

Here, x represents states including position, velocity, and acceleration, w means the zero-mean white Gaussian noise. And τ, I_3 indicate the time constant of acceleration and identity matrix of order 3, respectively. For filter design, the following discrete-time equivalent is needed.

$$\boldsymbol{x}_k = \Phi_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}, \text{ with } \boldsymbol{w}_{k-1} \sim N(0, Q_k). \tag{8}$$

$$\Phi_k \simeq I + F\Delta t \tag{9}$$

$$Q_k \simeq S_w Q_0 = S_w \begin{bmatrix} \frac{\Delta t^5}{20} I_3 & \frac{\Delta t^4}{8} I_3 & \frac{\Delta t^3}{6} I_3 \\ \frac{\Delta t^4}{8} I_3 & \frac{\Delta t^3}{3} I_3 & \frac{\Delta t^2}{2} I_3 \\ \frac{\Delta t^3}{6} I_3 & \frac{\Delta t^2}{2} I_3 & \Delta t I_3 \end{bmatrix} \tag{10}$$

where S_w is power spectral density and S_w and Q_0 is white noise jerk model. For the integral of the jerk over a time step Δt, the acceleration increment is represented in Eq. (10).

There are three types of information that can be measured by the radar system: elevation, azimuth, and range. Three measurements can be calculated using the relative position between a target and radar system.

$$[r_x \quad r_y \quad r_z]^T = [p_x \quad p_y \quad p_z]^T - [p_x^{rd} \quad p_y^{rd} \quad p_z^{rd}]^T \tag{11}$$

where the superscript rd denotes radar system and the state r represents the relative position between a target and radar system. Based on the relative information, two angle measurements and one range measurement can be described as follows.

$$\begin{bmatrix} z_\theta \\ z_\psi \\ z_R \end{bmatrix} = \begin{bmatrix} tan^{-1}\left(r_z \Big/ \sqrt{r_x^2 + r_y^2}\right) + n_{G,\theta} + n_\theta \\ tan^{-1}\left(r_y / r_x\right) + n_{G,\psi} + n_\psi \\ \sqrt{r_x^2 + r_y^2 + r_z^2} + n_R \end{bmatrix} \tag{12}$$

where z_θ, z_ψ and z_R are elevation, azimuth and range measurement, respectively. And as main error components, the receiver noises of radar system(n_θ, n_ψ, and n_R) and non-Gaussian glint noises($n_{G,\theta}$ and $n_{G,\psi}$) are generated [26].

B. THE LIMITATION OF REAL-TIME IMPLEMENTATION

Particle filter algorithm has been widely used due to its high estimation accuracy and consistency [29]. However, for high accuracy, a large number of particles are required, which exponentially increases the computation burden. However, against high-speed targets such as ballistic missile, the target estimation should be executed in real time. To cope with the problem, we suggest FPGA-based acceleration method for particle filter application.

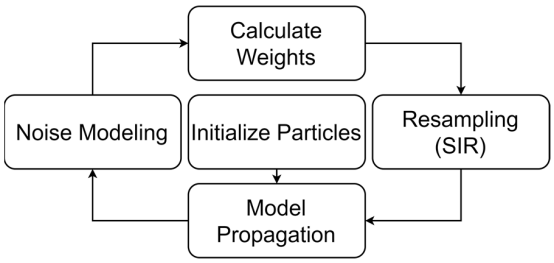


FIGURE 2. Flow diagram of our particle filter application illustrating the sequence of operations for ballistic target tracking.

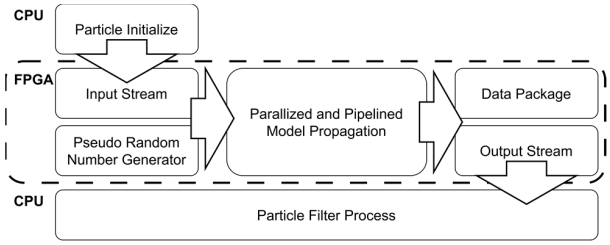


FIGURE 3. Overview of the proposed methods.

If the particle filter algorithm is executed on the CPU, all the iterative parts of a number of particle are computed sequentially. On the other hand, the repetitive calculation caused by many particles can be parallelized in the FPGA. Therefore, a considerable amount of computation burden can be greatly reduced when we adopt FPGA approach for particle filter algorithm. Although additional overhead time is required for FPGA approach, an appropriate parallelization technique should be suggested considering the computational efficiency.

III. PROPOSED METHOD

Fig. 2 illustrates the flowchart of our application for ballistic target tracking. As previously mentioned, by configuring a system to operate efficiently using the characteristics and resources of a limited system, it is possible to expect an increase in real-time performance due to a reduction in execution time.

Therefore, to accelerate the particle filter, we propose a system that parallelizes the algorithm for computation. Fig. 3 illustrates the structure of the proposed system, which incorporates the following methods for hardware latency and heterogeneous processor intercommunication for data exchange.

A. PARALLELIZATION AND PIPELINING

In our endeavor to minimize latency, a comprehensive analysis was conducted on the spatiotemporal complexity of the algorithm. This process led us to apply a strategic combination of pipelining and parallelization techniques that aimed to enhance the efficiency of the system. Upon profiling the execution time, it was evident that the section requiring the most computational time was the model propagation, as despite in Fig. 4.

TABLE 2. Pseudo code of baseline particle filter.

Algorithm 1 Particle Filter for Ballistic Target Tracking

N: **number of particles**
z_m: **measurements**
z_{bar}: **predicted measurements**

1: $x_p \leftarrow$ **Initialize Particles**
2: $x_{bar} \leftarrow$ **Model Propagation**
3: $n_\theta, n_\psi, n_R, n_{G,\theta}, n_{G,\psi} \leftarrow$ **Noise Modeling**
4: $p_\omega \leftarrow$ **Calculate Weights for particles**
for i **in range** (1:N):
5: $p_\rho \leftarrow \frac{1}{\sqrt{2\pi\sigma_\rho^2}} \exp\left(-\frac{(z_m[0]-z_{bar}[0][i])^2}{2\sigma_\rho^2}\right)$
6: $p_\theta \leftarrow \frac{1}{\sqrt{2\pi\sigma_\theta^2}} \exp\left(-\frac{(z_m[1]-z_{bar}[1][i])^2}{2\sigma_\theta^2}\right)$
7: $p_\psi \leftarrow \frac{1}{\sqrt{2\pi\sigma_\psi^2}} \exp\left(-\frac{(z_m[2]-z_{bar}[2][i])^2}{2\sigma_\psi^2}\right)$
8: $p_\omega[i] \leftarrow p_\rho \times p_\theta \times p_\psi$
9: $p_{\omega_{sum}} \leftarrow p_{\omega_{sum}} + p_\omega[i]$
10: **end**
11: **for** i **in range** (1:N):
12: $p_\omega[i] \leftarrow \frac{p_\omega[i]}{p_{\omega_{sum}}}$
13: **end**
14: $x_p \leftarrow$ **Sequential Importance Resampling (SIR)**
for i **in range** (1:N):
15: Random value generation
16: Combining cumulative weights with random values
17: Pair creation and sorting
18: Compute resampled index
19: $x_p \leftarrow$ Update(x_{bar})
20: **end**

In Table 2, line 17 refers to the sorting process within the Sequential Importance Resampling (SIR) method [30], and line 9 refers to the cumulative summation operation during weight calculation. Both operations, especially the sorting process and the cumulative summation, are not inherently conducive to parallel computation due to their intrinsic data dependencies. Additionally, when entering parallel processing, the I/O overhead can be significant. Hence, it's imperative to exercise careful consideration and discretion when determining which operations are viable candidates for acceleration. However, model propagation involves a vast number of independent computations, making it worthwhile to pursue parallel processing despite the I/O overheads.

Therefore, to enhance efficiency, the model propagation procedure, typically consisting of addition and multiplication tasks on a 9×9 matrix and a 9×1 matrix, underwent a redefinition. As depicted in Fig. 5, the 9×9 matrix

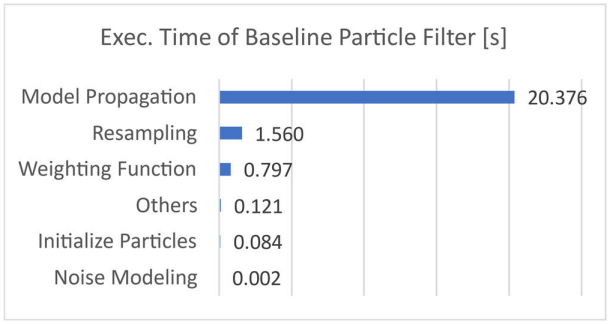


FIGURE 4. Profiling results of the particle filter application execution time, highlighting the model propagation step as the most time intensive. The figure emphasizes the suitability of parallelization and pipelining for the Model Propagation step due to its nature of performing multi-dimensional matrix product with large continuity.

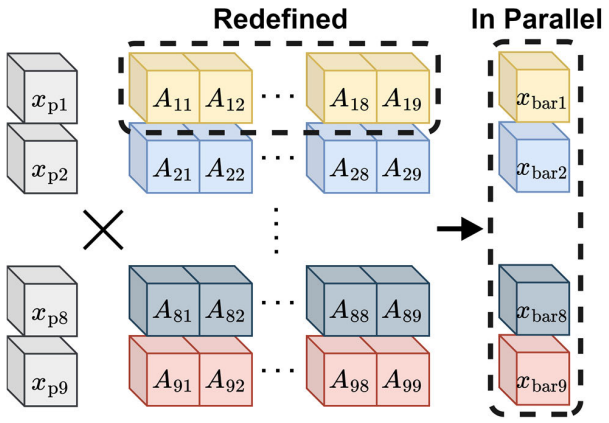


FIGURE 5. Efficiency optimization methods in matrix operations. Redefinition of the 9 × 9 filter covariance and motion model matrices enable efficient parallel computations in matrix multiplication.

was ingeniously split into separate rows, thereby facilitating parallel computations within the system. In circumstances where data dependencies were unavoidable, pipeline methods were integrated to further optimize the process and reduce latency.

B. A SHIFT TOWARDS LIGHTWEIGHT LFSR ALGORITHMS

Various algorithms for pseudorandom number generation exist, with the Mersenne Twister algorithm being the most prevalently used [23]. Despite this, its high computational complexity often renders it unsuitable for real-time operations and maintaining responsiveness in embedded systems. Table 3 shows our lightweight pseudo random number generator. In the context of particle filtering, the random numbers employed are not purposed for cryptography, thus paving the way for lightweight pseudorandom number generation. This can be accomplished by suitably integrating a seed into algorithms such as the Linear Feedback Shift Register (LFSR). In our research, the capability of a 32-bit LFSR pseudorandom number generation algorithm has been harnessed.

The measurement serves as the seed input, as shown in Fig. 6. Given that particle filtering necessitates normally distributed random numbers, and the Box-Muller transformation integrated to generate the required random numbers.

TABLE 3. Pseudo code of proposed random number generator.

Algorithm 2 Pseudo Random Number Generator

INTPUT: $seed$

OUTPUT: $\boldsymbol{rand}_n$

Initialization of variables: assign number **i**

While (i < random numbers) do

  $lfsr_a \leftarrow \text{LFSR}(seed)$ // 32bit random number

  $lfsr_b \leftarrow \text{LFSR}(lfsr_a)$ // 32bit random number

  $rand_a, rand_b \leftarrow \text{dist}(lfsr_a, lfsr_b) \sim U(0,1)$

  $\boldsymbol{rand}_n \leftarrow \text{box-muller}(rand_a, rand_b) \sim N(0,1)$

end

return $\boldsymbol{rand}_n$

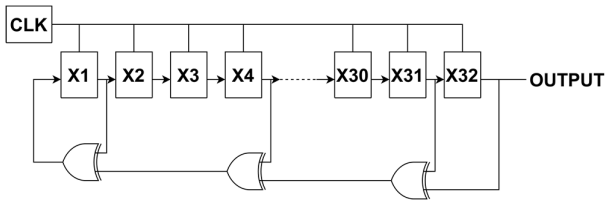


FIGURE 6. Schematic of the 32-bit LFSR used for random number generation.

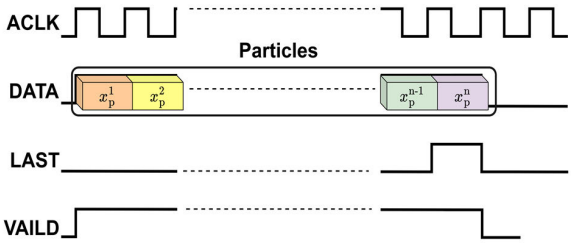


FIGURE 7. An illustration of the implemented stream-like protocol is shown, enabling parallel processing of multiple particles, with a last signal activation denoting the end of the stream.

C. STREAM-LIKE DATA EXCHANGE PROTOCOL

The need to design a data exchange protocol between the CPU and FPGA is a critical aspect of our work. This can be implemented using a structure that is reminiscent of a stream-like format, owing to its parallelism. The choice of structure largely depends on the volume of data that needs to be exchanged. For instance, a simple structure is apt when there is a requirement to exchange large amounts of data at once. On the other hand, a stream-like structure is more appropriate when the data exchange needs are less intensive. Given that our research involves parallel processing, which necessitates the exchange of significant amounts of data concurrently, a stream-like structure is deemed most suitable, as depicted in Fig. 7.

IV. EVALUATION RESULTS

The setup for the system environment used in our research is as follows: The Xilinx ZYNQ ZC706 evaluation board was used for testing purposes, and the Xilinx Vitis tool,

TABLE 4. Resource utilization of the proposed method on FPGA.

Particles	8	16	32	64	128
LUT	22291	24433	29297	31844	52222
LUTRAM	823	833	841	1022	1020
FF	28442	33195	42584	61327	97105
BRAM	5	5	5	5	5
DSP	192	192	192	192	192
BUFG	1	1	1	1	1
POWER	2.194W	2.247W	2.273W	2.372W	2.537W

a prominent software platform for accelerating applications, was employed for co-simulation. When the proposed method was implemented, the utilization of FPGA resources was as presented in Table 4. In this table, "Particles" refers to the number of particles processed in parallel in the implemented hardware.

A. RESULTS OF HIGH-SPEED TARGET TRACKING

A tracking scenario for a ballistic missile is proposed to verify the effectiveness of the proposed acceleration method. For the target trajectory, the dynamic equations in Eq. (1) to Eq. (6) are used and missile specification including aerodynamic drag were set as in [27]. The sampling interval and total simulation time are set to be 0.01second and 2second yielding 200 intervals. The standard deviation of the radar receiver noises were set to be $0.1°(n_\theta)$, $0.1°(n_\psi)$, and $1\text{m}(n_R)$, respectively. Glint noises such as $n_{G,\theta}$ and $n_{G,\psi}$ are mixtures of Gaussians as follows.

$$p(x) = (1-\varepsilon)\, p_{G_1} + \varepsilon p_{G_2}. \tag{13}$$

where ε is the glint probability and is set to be 0.3. p_{G_1} and p_{G_2} are Gaussians in $p_{G_1} \sim N(0, 0.5^2)°$ and $p_{G_2} \sim N(0, 5^2)°$ at the range of 100m in respectively [26]. Under the general assumption that the radar is fixed on the ground, the Singer model is used for target tracking. The velocity of a ballistic missile varies with time due to the gravitational force and aerodynamic drag, which is well shown in Fig. 8. The number of particles for tracking filter is set to be 15000. It can be shown that although the initial estimation is very inaccurate, the tracking filter follows the true value in a short time. In Fig. 9 and Fig. 10, the estimation results for position appear, which shows a satisfactory tracking performance.

B. PERFORMANCE ANALYSIS OF PROPOSED METHOD

1) PIPELINING AND PARALLELIZE

Fig. 11 provides a detailed depiction of the timing diagram, representing the hardware implementation that was derived from the proposed methodology. During the implementation process, data dependencies were carefully considered, and pipelining was applied specifically to the prediction operation for each individual particle to ensure that the operation was carried out efficiently. However, it's important to note that the process of data exchange with the CPU, as visualized in the diagram, introduces an inherent overhead. While this overhead is unavoidable, it is a necessary part of the system's operation, facilitating the crucial communication between the CPU and other hardware components.

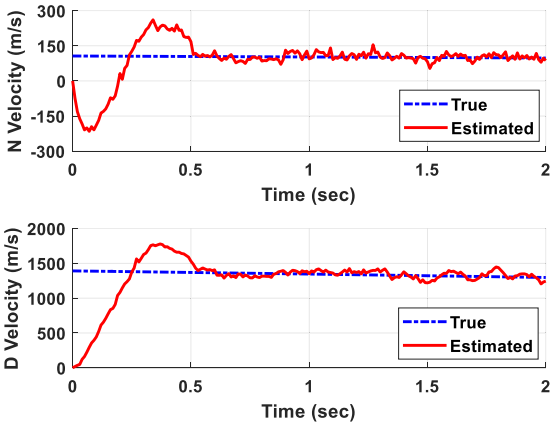


FIGURE 8. Estimating target velocity compared to true velocity.

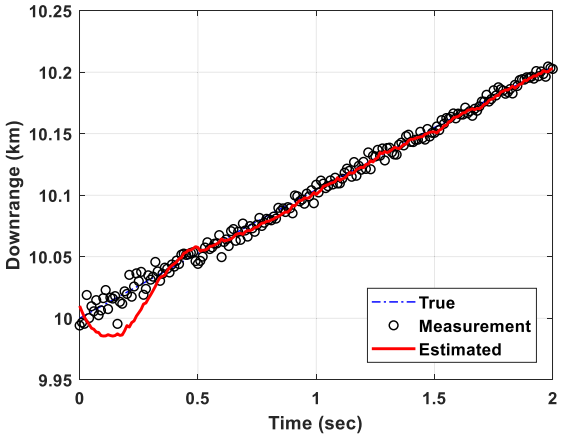


FIGURE 9. Estimating target downrange compared to measurement and true value.

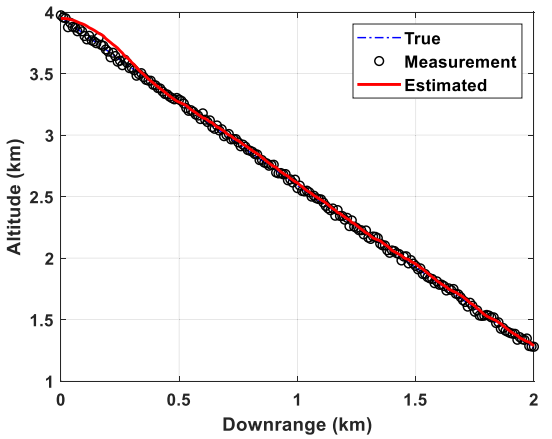


FIGURE 10. Estimating target altitude compared to measurement and true value.

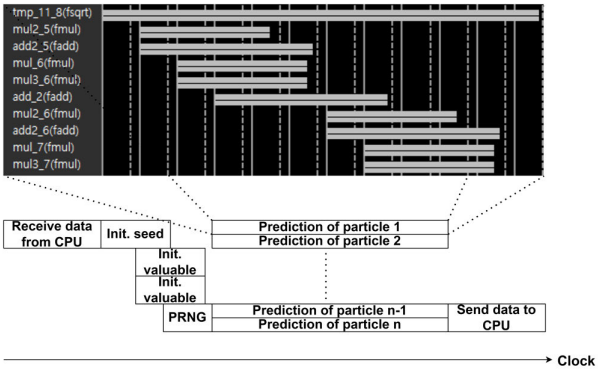


FIGURE 11. Results of the implemented pipeline and a timing diagram are presented, demonstrating efficient execution of operations on multiple particles within a single task.

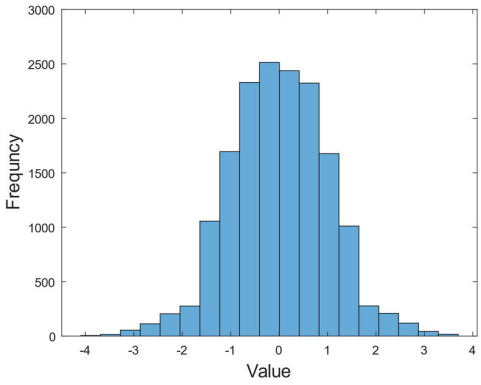


FIGURE 12. Results of random number generation using our proposed method.

2) RESULTS OF GENERATE GAUSSIAN RANDOM NUMBERS

Fig. 12 depicts a histogram of 2^{14} numbers that were randomly generated by the proposed method. The adherence of these generated numbers to a suitable standard normal distribution, crucial for their use as inputs in the particle filter, is demonstrated. Moreover, the ease of implementation and efficiency in terms of resource usage are attributes associated with the LFSR structure, making it advantageous in terms of hardware implementation.

3) COMPARISON IN COMPUTATION TIME OF VARIOUS RANDOM NUMBER GENERATOR

In Table 5, a comparison of execution times is presented, featuring the Mersenne Twister (MT19937), the 64-bit Mersenne Twister (MT19937-64bit) pseudo-random number generators, and the method proposed, within the context of an Intel Core i9-12900K desktop environment. A period of 2^{19937}-1 is exhibited by the Mersenne Twister algorithm, which is much longer compared to the period of 2^{32}-1 of the proposed method. Although the period of the Mersenne Twister algorithm is difficult to predict, a sufficient period for our application is still provided. A clear demonstration of the efficiency of the proposed algorithm is provided by the comparison, as it shows approximately twice the speed in terms of execution time compared to the other two algorithms. These results demonstrate that the numbers generated by the proposed pseudo-random number generation algorithm not only adhere to a suitable standard normal distribution, but they also do so more efficiently.

TABLE 5. Execution time of various random number generators.

Method	Exec. Time (s)
Proposed	**2.726±0.086**
MT19937	7.229±0.186
MT19937-64bit	8.049±0.872

4) COMPUTATION TIME ACCORDING TO THE NUMBER OF PARTICLES

In Table 6, the overall execution time of the algorithm, based on the number of particles, is provided. The most significant speed improvement is shown to result from the application of pipelining. Additionally, speed enhancement is also found to be contributed by the sequential co-design approach. However, when all proposed methods are applied and the number of particles processed in parallel is increased, a bottleneck is observed. In conclusion, significant speed improvement is exhibited by the algorithm, particularly on a hardware-software co-design system, where the total execution time is seen to be approximately four times faster than the baseline. This substantial improvement is attributed to the successful application of parallelization, pipelining, and data processing techniques. Considering FPGA resource usage and power consumption, a particle filter system suitable for specific objectives can be selected based on these results.

C. DISCUSSION

Our experiments with various hardware implementation techniques and hardware-software (HW/SW) co-design approaches were conducted, showing their substantial contribution to system acceleration. This is particularly evident in the application of particle filtering for ballistic target tracking, where the necessity of reducing execution time is highlighted.

The hardware implementation section was focused on pipelining, parallelization, and array redefinition. Among these, pipelining was found to be the most impactful, leading to a significant reduction in hardware latency. Importantly, when all hardware acceleration methods were collectively applied, a substantial speedup was achieved by the system, exceeding a factor of 33. In the HW/SW co-design section, methods including stream-like protocols and pseudo-random number generation algorithms were explored. Again, significant contribution to the reduction of execution time was observed from all methods, achieving a speedup of approximately four times when used collectively. These findings underscore the crucial role of hardware and software co-design in the optimization of particle filters for

TABLE 6. Comparison of total execution time of particle filter according to the number of particles and the proposed method.

Number of Particles	Method	Exec. Time (s) according to the number of particles processed in parallel				
		8 Particles	16 Particles	32 Particles	64 Particles	128 Particles
1024	Only CPU			22.940 (for all cases)		
	CPU+FPGA (Only pipelining)			6.392 (for all cases)		
	CPU+FPGA (Sequential)	14.677	11.299	9.618	8.771	8.355
	CPU+FPGA (Proposed)	4.942	4.822	4.768	4.738	4.729
2048	Only CPU			45.947 (for all cases)		
	CPU+FPGA (Only pipelining)			13.254 (for all cases)		
	CPU+FPGA (Sequential)	29.820	23.064	19.702	18.008	17.177
	CPU+FPGA (Proposed)	10.351	10.111	10.002	9.941	9.924
3072	Only CPU			69.486 (for all cases)		
	CPU+FPGA (Only pipelining)			19.485 (for all cases)		
	CPU+FPGA (Sequential)	44.344	34.210	29.167	26.627	23.380
	CPU+FPGA (Proposed)	15.141	14.781	14.617	14.527	14.500
4096	Only CPU			92.674 (for all cases)		
	CPU+FPGA (Only pipelining)			25.694 (for all cases)		
	CPU+FPGA (Sequential)	58.840	45.328	38.604	35.216	33.554
	CPU+FPGA (Proposed)	19.902	19.423	19.204	19.084	19.048
5120	Only CPU			115.726 (for all cases)		
	CPU+FPGA (Only pipelining)			32.851 (for all cases)		
	CPU+FPGA (Sequential)	74.270	57.381	48.975	44.472	42.662
	CPU+FPGA (Proposed)	25.598	24.999	24.725	24.575	24.531

real-time operation in ballistic target tracking systems. The proposed methods are a significant advancement in this respect, demonstrating that efficient management of power consumption and significant acceleration can be concurrently achieved.

V. CONCLUSION

This paper has tackled the significant challenge of real-time tracking of ballistic targets using sampling-based particle filters. The high performance of these filters in tracking non-Gaussian targets is established, yet their accuracy is inherently tied to the number of particles utilized. This creates a computational demand that, along with the spatial and power constraints of an airborne platform, poses a complex challenge. To address this, a hardware-software co-design strategy was proposed and implemented, accelerating the particle filter algorithm, and facilitating parallel processing while optimizing energy utilization.

Experimental evaluations validated the effectiveness of the proposed approach, which resulted in over four times acceleration, demonstrating a significant advance towards real-time and efficient ballistic target tracking. This success highlights the potential of hardware-software co-design strategies in overcoming computational and resource constraints while maintaining, and even enhancing, algorithmic performance. The results suggest that the proposed solution holds promise for further research and development in the field of real-time target tracking and similar high-demand applications.

REFERENCES

[1] J. Kim, S. S. Vaddi, P. K. Menon, and E. J. Ohlmeyer, "Comparison between nonlinear filtering techniques for spiraling ballistic missile state estimation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 1, pp. 313–328, Jan. 2012, doi: 10.1109/TAES.2012.6129638.

[2] A. Farina, B. Ristic, and D. Benvenuti, "Tracking a ballistic target: Comparison of several nonlinear filters," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 38, no. 3, pp. 854–867, Jul. 2002, doi: 10.1109/TAES.2002.1039404.

[3] C. Moon, J. Tak, S.-H. Kim, and K. Song, "Ballistic coefficient estimation with Gaussian process particle filter," in *Proc. 18th Int. Conf. Control, Autom. Syst. (ICCAS)*, Oct. 2018, pp. 776–780.

[4] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002, doi: 10.1109/78.978374.

[5] C. Kwok, D. Fox, and M. Meila, "Real-time particle filters," *Proc. IEEE*, vol. 92, no. 3, pp. 469–484, Mar. 2004, doi: 10.1109/JPROC.2003.823144.

[6] G. Hendeby, R. Karlsson, and F. Gustafsson, "Particle filtering: The need for speed," *EURASIP J. Adv. Signal Process.*, vol. 2010, no. 1, pp. 1–9, Dec. 2010.

[7] D. Kim, Y. Han, H. Lee, Y. Kim, H.-H. Kwon, C. Kim, and W. Choi, "Accelerated particle filter with GPU for real-time ballistic target tracking," *IEEE Access*, vol. 11, pp. 12139–12149, 2023, doi: 10.1109/ACCESS.2023.3238873.

[8] R. Cabido, A. S. Montemayor, and J. J. Pantrigo, "High performance memetic algorithm particle filter for multiple object tracking on modern GPUs," *Soft Comput.*, vol. 16, no. 2, pp. 217–230, Feb. 2012.

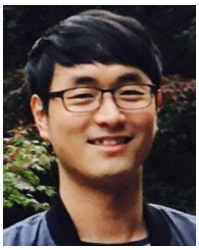
[9] G. Szwoch, "Performance evaluation of the parallel object tracking algorithm employing the particle filter," in *Proc. Signal Process., Algorithms, Architectures, Arrangements, Appl.*, 2016, pp. 119–124.

[10] X. Tang and Z. Fu, "CPU–GPU utilization aware energy-efficient scheduling algorithm on heterogeneous computing systems," *IEEE Access*, vol. 8, pp. 58948–58958, 2020, doi: 10.1109/ACCESS.2020.2982956.

[11] A. Jarrah, M. M. Jamali, and S. S. S. Hosseini, "Optimized FPGA based implementation of particle filter for tracking applications," in *Proc. IEEE Nat. Aerosp. Electron. Conf.*, Jun. 2014, pp. 233–236, doi: 10.1109/NAECON.2014.7045808.

[12] S. Agrawal, P. Engineer, R. Velmurugan, and S. Patkar, "FPGA implementation of particle filter based object tracking in video," in *Proc. Int. Symp. Electron. Syst. Design (ISED)*, Kolkata, India, 2012, pp. 82–86, doi: 10.1109/ISED.2012.41.
[13] A. Tahara, Y. Hayashida, T. T. Thu, Y. Shibata, and K. Oguri, "FPGA-based real-time object tracking using a particle filter with stream architecture," in *Proc. 4th Int. Symp. Comput. Netw. (CANDAR)*, Hiroshima, Japan, Nov. 2016, pp. 422–428, doi: 10.1109/CANDAR.2016.0079.
[14] A. Jarrah, M. M. Jamali, S. S. S. Hosseini, J. Astola, and M. Gabbouj, "Parralelization of non-linear & non-Gaussian Bayesian state estimators (particle filters)," in *Proc. 23rd Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2015, pp. 2506–2510, doi: 10.1109/EUSIPCO.2015.7362836.
[15] Y. M. Wijesinghe, J. G. Samarawickrama, and D. Dias, "Hardware and software co-design for object detection with modified ViBe algorithm and particle filtering based object tracking," in *Proc. 14th Conf. Ind. Inf. Syst. (ICIIS)*, Kandy, Sri Lank, Dec. 2019, pp. 506–511, doi: 10.1109/ICIIS47346.2019.9063249.
[16] M. Sadasivam and S. Hong, "Application specific coarse-grained FPGA for processing element in real time parallel particle filters," in *Proc. 3rd IEEE Int. Workshop Syst.-Chip Real-Time Appl.*, Calgary, AB, Canada, Jul. 2003, pp. 116–119, doi: 10.1109/IWSOC.2003.1213018.
[17] S. Liu, G. Mingas, and C.-S. Bouganis, "Parallel resampling for particle filters on FPGAs," in *Proc. Int. Conf. Field-Programmable Technol. (FPT)*, Shanghai, China, Dec. 2014, pp. 191–198, doi: 10.1109/FPT.2014.7082775.
[18] Y. Song, D. Liu, and Y. Peng, "FPGA-based implementation of lithium-ion battery SOH estimator using particle filter," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (IMTC)*, Dubrovnik, Croatia, May 2020, pp. 1–6, doi: 10.1109/I2MTC43012.2020.9128439.
[19] H. A. Abd El-Halym, I. I. Mahmoud, and S. E.-D. Habib, "Efficient hardware architecture for particle filter based object tracking," in *Proc. IEEE Int. Conf. Image Process.*, Hong Kong, Sep. 2010, pp. 4497–4500, doi: 10.1109/ICIP.2010.5653817.
[20] A. Rodríguez and F. Moreno, "Evolutionary computing and particle filtering: A hardware-based motion estimation system," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3140–3152, Nov. 2015, doi: 10.1109/TC.2015.2401015.
[21] L. Miao, J. J. Zhang, C. Chakrabarti, and A. Papandreou-Suppappola, "Efficient Bayesian tracking of multiple sources of neural activity: Algorithms and real-time FPGA implementation," *IEEE Trans. Signal Process.*, vol. 61, no. 3, pp. 633–647, Feb. 1, 2013, doi: 10.1109/TSP.2012.2226172.
[22] T. Chisholm, R. Lins, and S. Givigi, "FPGA-based design for real-time crack detection based on particle filter," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 5703–5711, Sep. 2020, doi: 10.1109/TII.2019.2950255.
[23] Y. Li, P. Chow, J. Jiang, and M. Zhang, "Software/hardware framework for generating parallel long-period random numbers using the WELL method," in *Proc. 21st Int. Conf. Field Program. Log. Appl.*, Chania, Greece, Sep. 2011, pp. 110–115, doi: 10.1109/FPL.2011.29.
[24] R. A. Singer, "Estimating optimal tracking filter performance for manned maneuvering targets," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-6, no. 4, pp. 473–483, Jul. 1970, doi: 10.1109/TAES.1970.310128.
[25] X. Rong Li and V. P. Jilkov, "Survey of maneuvering targettracking. Part I: Dynamic models," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1333–1364, Oct. 2003, doi: 10.1109/TAES.2003.1261132.
[26] J. Kim, M. Tandale, P. K. Menon, and E. Ohlmeyer, "Particle filter for ballistic target tracking with glint noise," *J. Guid., Control, Dyn.*, vol. 33, no. 6, pp. 1918–1921, Nov. 2010, doi: 10.2514/1.51000.
[27] D. C. Wright and T. Kadyshev, "An analysis of the north Korean nodong missile," *Sci. Global Secur.*, vol. 4, no. 2, pp. 129–160, Feb. 1994, doi: 10.1080/08929889408426397.
[28] B. Wang, J. Xie, S. Li, Y. Wan, Y. Gu, S. Fu, and K. Lu, "Computing in the air: An open airborne computing platform," *IET Commun.*, vol. 14, no. 15, pp. 2410–2419, Sep. 2020, doi: 10.1049/iet-com.2019.0515.
[29] S. Thrun, "Particle filters in robotics," in *Proc. 18th Conf. Uncertainty Artif. Intell.* San Francisco, CA, USA: Morgan Kaufmann Publishers, 2002, pp. 511–518.
[30] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 70–86, May 2015, doi: 10.1109/MSP.2014.2330626.

DAEYEON KIM received the B.S. degree in electronic engineering from the Kumoh National Institute of Technology, Gumi, Gyeongsangbuk-do, South Korea, in 2023, where he is currently pursuing the M.S. degree with the Department of IT Convergence Engineering. His current research interests include deep learning, real-time embedded systems, and algorithm acceleration with GPU and FPGA



HEONCHOEL LEE (Member, IEEE) received the B.S. degree in electronic engineering and computer sciences from Kyungpook National University, Daegu, South Korea, in 2006, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from Seoul National University, Seoul, South Korea, in 2008 and 2013, respectively. From 2013 to 2019, he was a Senior Researcher with the Agency for Defense Development, Daejeon, South Korea. Since 2019, he has been an Assistant Professor with the School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, South Korea. He is currently a Technical Adviser with the Robot Navigation Division, Cleaning Science Research Institute, LG Electronics. His current research interests include SLAM, robot navigation, machine learning, real-time embedded systems, prognostics, and health management.

HYUCK-HOON KWON received the B.S., M.S., and Ph.D. degrees in aerospace engineering from the Korea Advanced Institute of Science and Technology (KAIST), Deajeon, South Korea, in 2002, 2005, and 2020, respectively.

In 2009, he joined LIG Nex1, for the development of precision-guided missiles. His current research interests include convex optimization, optimal control, guidance and autopilot design, and nonlinear control.

YEJI HWANG received the B.S. degree in electronic engineering from Kyung Hee University, Yongin, South Korea, in 2020, and the M.S. degree in aerospace engineering from Inha University, Incheon, South Korea, in 2023.

Since 2023, she has been a Researcher with LIG Nex1, Seongnam, South Korea. Her current research interests include convex optimization, the guidance and control of unmanned aerial vehicles, and missile systems.

WONSEOK CHOI received the M.S. degree in defense convergence engineering from Yonsei University, Seoul, South Korea, in 2017.

From 2015 to 2017, he was a Senior Researcher with LIGNex1 for PGM R&D Group Development, Gyeonggi-do, South Korea. His current research interests include embedded systems, embedded SW, real-time embedded systems, and missile systems.