e-Prime - Advances in Electrical Engineering, Electronics and Energy 6 (2023) 100377

Contents lists available at ScienceDirect

e-Prime - Advances in Electrical Engineering, Electronics and Energy

journal homepage: www.elsevier.com/locate/prime

FPGA Implementation of Metaheuristic Optimization Algorithm

Nurul Hazlina Noordin [a,*], Phuah Soon Eu [a], Zuwairie Ibrahim [b]

[a] UMP STEM Lab, Faculty of Electrical & Electronics Engineering, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia
[b] Faculty of Manufacturing & Mechatronic Engineering, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia

ARTICLE INFO

Keywords:
FPGA design
Binary simulated Kalman filter

ABSTRACT

Metaheuristic algorithms are gaining popularity amongst researchers due to their ability to solve nonlinear optimization problems as well as the ability to be adapted to solve a variety of problems. There is a surge of novel metaheuristics proposed recently, however it is uncertain whether they are suitable for FPGA implementation. In addition, there exists a variety of design methodologies to implement metaheuristics upon FPGA which may improve the performance of the implementation. The project begins by researching and identifying metaheuristics which are suitable for FPGA implementation. The selected metaheuristic was the Simulated Kalman Filter (SKF) which proposed an algorithm that was low in complexity and used a small number of steps. Then the Discrete SKF was adapted from the original metaheuristic by rounding all floating-point values to integers as well as setting a fixed Kalman gain of 0.5. The Discrete SKF was then modeled using behavioral modeling to produce the Binary SKF which was then implemented onto FPGA. The design was made modular by producing separate modules that managed different parts of the metaheuristic and Parallel-In-Parallel-Out configuration of ports was also implemented. The Discrete SKF was then simulated on MATLAB meanwhile the Binary SKF was implemented onto FPGA and their performance were measured based on chip utilization, processing speed, and accuracy of results. The Binary SKF produced speed increment of up to 69 times faster than the Discrete SKF simulation.

Introduction

Metaheuristic algorithms have emerged as the bedrock of modern problem-solving techniques, offering the flexibility to tackle a wide range of optimization challenges. It is an optimization method that is used to solve complex nonlinear and multimodal problems [1]. These algorithms draw inspiration from various natural and computational processes, adapting and evolving to provide innovative approaches to problem-solving. Within the topic of metaheuristics, the Simulated Kalman Filter (SKF) offers an edge of simple execution in addressing optimization and adaptability [1–3]. Its applications span across domains as diverse as supply chain management, finance, wireless sensor networks, and engineering design. SKF's ability to swiftly adapt to real-time optimization tasks and deliver promising results makes it a formidable candidate for addressing complex, real-world problems.

As the demand for robust optimization solutions intensifies, so does the need for advancements in implementation methods. Among these methods, the implementation of metaheuristic algorithms on Field Programmable Gate Arrays (FPGAs) has gained considerable attention. FPGAs offer the promise of unparalleled processing speed and efficiency, potentially revolutionizing the way metaheuristic algorithms are executed. Nevertheless, the implementation of complex metaheuristics on FPGAs comes with its own set of intricacies and challenges. This study embarks on a comprehensive exploration of the Binary Simulated Kalman Filter (Binary SKF) algorithm and its FPGA implementation. The study delves deep into the intricacies of Binary SKF, dissecting its core components and principles, and elucidates its potential in addressing real-world optimization problems. The paper distinguishes SKF from alternative bio-inspired metaheuristics, laying the foundation for the decision to proceed with its FPGA implementation. Central to this academic exploration is a detailed examination of the SKF metaheuristic itself. The algorithm is dissected into its elemental components, each contributing significantly to its optimization capabilities. The details of population generation, fitness evaluation, and the Kalman Filter components are unraveled, presenting a holistic view of the algorithm's inner workings. A thorough comprehension of SKF is vital in understanding the intricacies of its FPGA implementation. The transition from the original SKF to a discrete version optimized for FPGA

* Corresponding author.
E-mail address: hazlina@umpsa.edu.my (N.H. Noordin).

https://doi.org/10.1016/j.prime.2023.100377
Received 2 July 2023; Received in revised form 5 November 2023; Accepted 24 November 2023
Available online 2 December 2023

implementation is scrutinized through MATLAB simulations. The paper highlighted the importance of this shift, primarily based on the adaptability of the discrete version to digital systems. This implementation shift enables greater precision and efficiency when implementing Binary SKF. Furthermore, this academic journal delves into the core of the Binary SKF FPGA implementation. The intricate process of behavioral modeling is meticulously examined, with a particular emphasis on the modules that underpin the FPGA implementation. Each module is individually designed, optimizing FPGA resources to ensure the algorithm's robust performance.

Crucial to this study is the exploration of speed improvements achieved through FPGA implementation. The paper seeks to identify the precise areas within Binary SKF where speed enhancements are most significant. Operations such as agent interaction, solution evaluation, control logic, data communication, resource allocation, and iteration management are scrutinized. The analysis shed light on the instrumental role played by FPGA technology in augmenting the performance and practical applicability of Binary SKF in a range of real-world optimization problems. The limitations and areas for improvement in the Binary SKF implementation are discussed, focusing on issues related to accuracy and precision, emphasizing the necessity for transitioning from a fixed Kalman gain to a dynamic adaptation model. Additionally, resource constraints and the potential vulnerability of the Binary SKF implementation to security threats are also discussed.

Additionally, the paper explores scalability, highlighting the fine balance between speed and resource utilization, particularly in applications with varying dimensions. Strategies to optimize resource sharing and ensure efficient utilization are touched upon. Moreover, this journal envisages the broader implications and applications of Binary SKF in diverse real-world scenarios. It highlights the versatility of the algorithm across industries and underscores the potential benefits it brings in terms of accuracy, efficiency, and adaptability. In pursuit of even greater accuracy and precision, the study concludes by introducing strategies aimed at enhancing the Binary SKF implementation. These strategies encompass the use of dynamic Kalman gain, precision enhancement through fixed-point arithmetic, hybridization with complementary optimization methods, and post-processing and refinement techniques.

Literature review

Novel optimization algorithms

The first objective of the project is to study different metaheuristic algorithms and to identify suitable metaheuristics for hardware implementation. The project begins by studying a variety of novel optimization algorithms.

Single-agent finite impulse response optimizer

The metaheuristic proposed is a single-agent metaheuristics that is inspired by the unbiased finite impulse response filter. It proposes an algorithm that optimizes a single solution iteratively until a stopping condition is met. It boasts a great performance in exploration and exploitation which enables it to search a wide range of possible solutions and lastly produce a near optimum solution [3]. Its process is described in the flowchart as shown in Fig. 1.

Barnacles mating optimization

The proposed algorithm is a novel multi-agent optimization algorithm which mimics the mating behaviors of barnacles as described in Fig. 2. It involves a sequence where barnacles are randomly selected, and the reproduction process occurs to a set population of barnacles. Then, the barnacles may only mate with the surrounding barnacles based on the length of their penis which is set prior to simulation. The offspring of the barnacles will then inherit the characteristics from its parents [4].

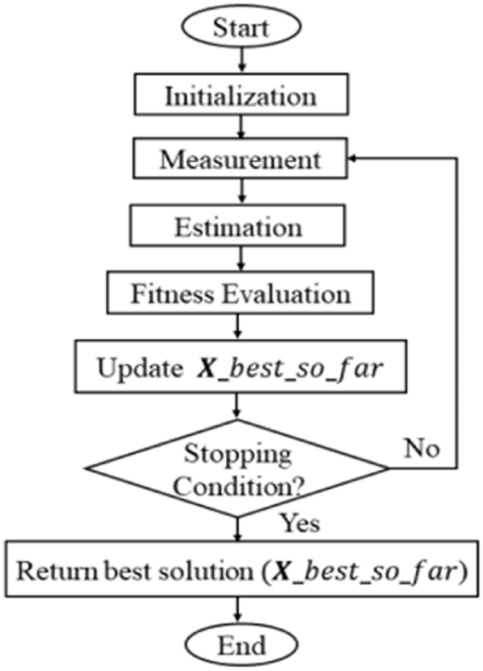

Fig. 1. Flowchart for single-agent finite impulse response optimizer.

Orca predation algorithm

A novel multi-agent bio-inspired metaheuristic which mimics the hunting behavior of orcas. The metaheuristic introduces a sequence where orcas drive, encircle, and attack a school of fish. The algorithm emphasizes on different stages of the sequence such as driving and encircling to effectively adjust its exploration and exploitation respectively. This enables the algorithm to solve a large variety of problems as it was implemented onto several engineering optimization problems which showed great performance [5]. The metaheuristic is described using a complex flowchart as seen in Fig. 3.

Simulated Kalman filter

A multi-agent metaheuristic where each search agent acts as a Kalman Filter which is a state estimation method popularized in the year 1960. Each search agent then estimates the optimum solution to the fitness function through several steps such as predict, measure, and estimate to consequently produce the best-so-far solution as shown in Fig. 4 [2].

Particle swarm optimization

A multi-agent metaheuristic inspired by the movement of flock of birds such as scattering and regrouping in search of food. Each agent also known as a particle is a candidate solution which moves around the search space during each iteration in search of improvements to the solution. The position and velocity are influenced by each particle's best-known position as well as the best-known position of other particles as well [6]. The flowchart for Particle Swarm Optimization is illustrated in Fig. 5.

Variants of particle swarm optimization

The Binary Particle Swarm Optimization was introduced to adapt the original algorithm into a discrete search space which overcomes the problems faced by the original algorithm which was designed to be used in a continuous search space. In this variant, the particles represent its position in binary meanwhile its velocity is defined as the probability that it will change its state. The structure of the variant is like the original algorithm however it utilizes a separate set of equations since it is adapted to work in binary [7].

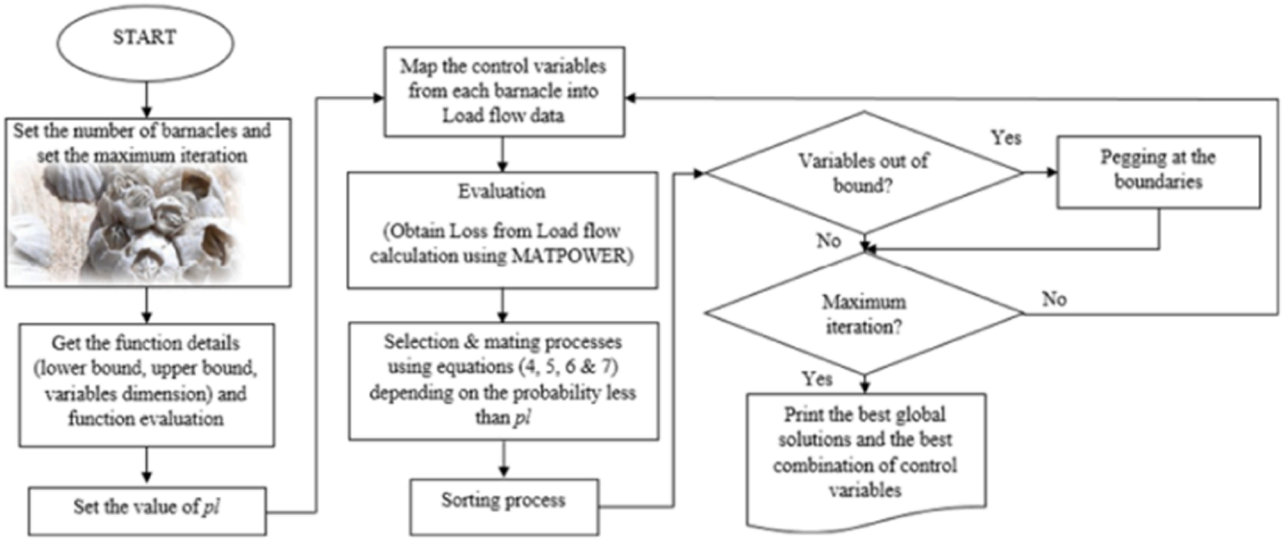

Fig. 2. Flowchart for barnacles mating optimizer.

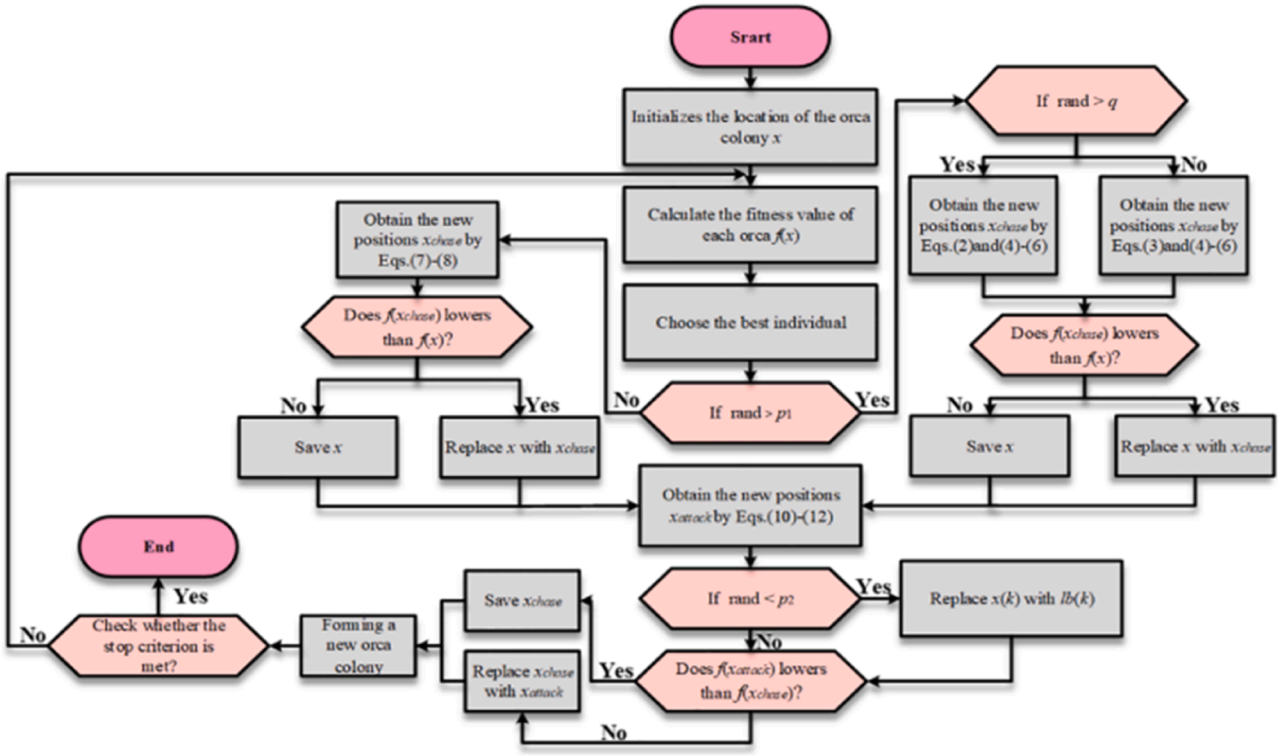

Fig. 3. Flowchart for orca predation algorithm.

Binary ant colony optimization

In the year 1991, the original Ant Colony Optimization metaheuristic was introduced. This paper implements the original metaheuristic into a binary solution domain such that the solution search space is represented in a binary format. Then its performance was verified through a binary function optimization problem opposed to the typical continuous function optimization problem [8].

Hybrid binary bat enhanced particle swarm optimization algorithm

This metaheuristic combines two variants of the original Bat optimization and Particle Swarm Optimization to form a hybrid metaheuristic namely a combination of binary Bat metaheuristic and binary Particle Swarm Optimization to form the Hybrid Binary Bat Enhanced Particle Swarm Optimization Algorithm. It is claimed that the binary variants of metaheuristics are capable of producing superior results.

The binary Bat metaheuristic applies a binary map onto the solution found since the solution search space is continuous. Meanwhile the Binary Particle Swarm Optimization converts continuous values into binary values. The results shows that the hybrid metaheuristic is capable of producing better results than other binary variants of other metaheuristics such as binary Genetic Algorithm, binary Particle Swarm Optimization, binary Greywolf, binary Bat, and binary Dragonfly. The hybrid combination of both binary Bat and binary Particle Swarm Optimization produced better solutions than the individual

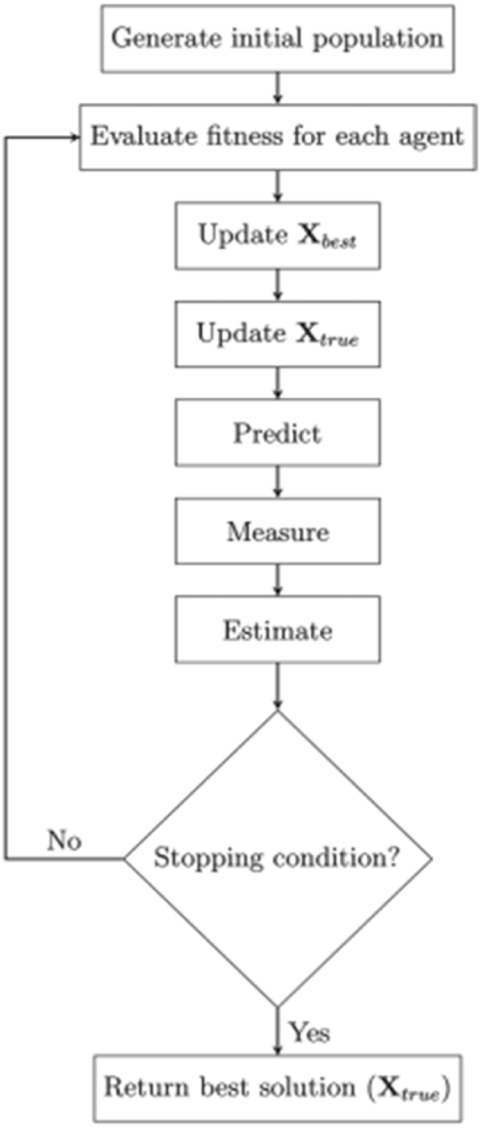

Fig. 4. Flowchart for simulated Kalman filter.

metaheuristics [9].

FPGA implementation optimization techniques

The performance of a metaheuristics on FPGA is dependent upon its design methodology. A typical implementation of a metaheuristic may vary in comparison to the implementation with different design methodology such as parallel implementations, pipeline architectures, and implementation of floating-point arithmetic modules.

Parallel implementation of particle swarm optimization

This implementation optimization technique was implemented to accelerate the processing speed of the algorithm. It was designed to process large volumes of data from processes such as Big Data and Mining of Massive Datasets. This was achieved through the implementation of several particle modules in parallel to concurrently compare the best fitness value of all particles as shown in Fig. 6. Despite the increment of particle modules to enable parallel computing of the particles, the results of the hardware costs showed that it used less registers and Look Up Tables in comparison to similar works from the literature [10].

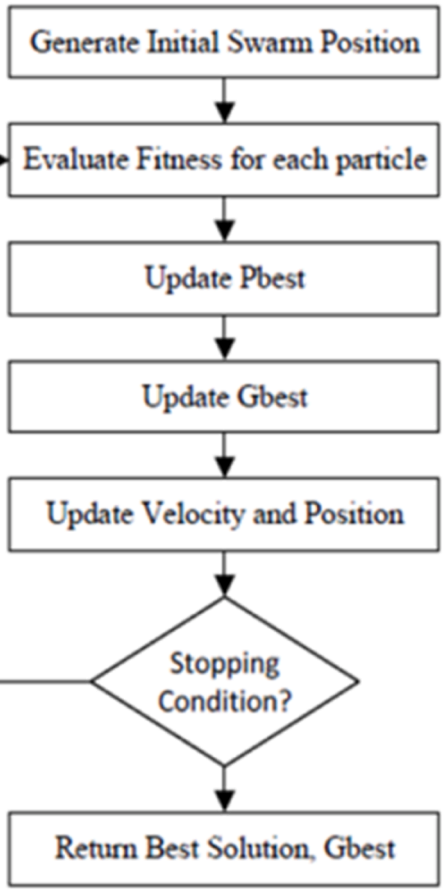

Fig. 5. Flowchart for particle swarm optimization.

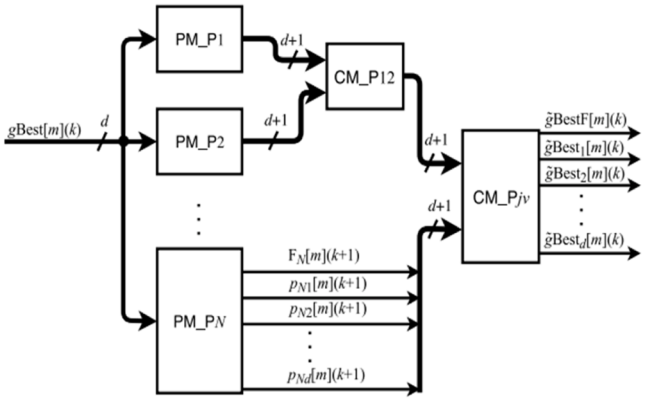

Fig. 6. Parallel Implementation of Particle Modules.

Pipeline architecture of particle swarm optimization

The paper works upon the previously mentioned variant of the Particle Swarm Optimization with random time-varying inertia weight and acceleration coefficients and its existing serial architecture implemented on hardware. The improvements proposed in this paper includes simplifications of the hardware architecture such as changing the control mechanism from a complicated distributed style into a simple centralized style to improve the stability of the hardware system as shown in Fig. 8. This was done to overcome the calculation error that occurred in the previous implementation. Next, the performance of the implementation was improved by introducing registers and reconfiguration of the state transitions.

The implementation was able to successfully run 6 concurrent

operations through a pipeline architecture as shown in Fig. 7. In addition, the simplification of hardware architecture was able to successfully reduce errors and improve the rate of available results. The results reported showed a significant improvement in terms of performance. However, an increment in chip usage was observed [11].

FPGA realization of particle swarm optimization algorithm using floating point arithmetic

The paper introduces the implementation of the original Particle Swarm Optimization algorithm with a modification upon its arithmetic modules. The arithmetic modules were replaced with floating-point arithmetic modules to further improve the accuracy of each particle results. This is because floating-point arithmetic used such as the single precision and double precision floating point arithmetic module enables the implementation to calculate to more decimal points. However, this resulted in extremely high chip usage. The implementation of the double precision floating point arithmetic module exceeded the total hardware available. The hardware cost was reduced by further implementing resource sharing for multiplication and addition operations which managed to free up some Look Up Tables [12].

Security implications of FPGA implementation

Security implications of FPGA implementation

The integration of field-programmable gate arrays (FPGAs) in various computational domains has witnessed substantial growth and innovation in recent years. While FPGA technology offers exceptional speed and flexibility, it is imperative to recognize that FPGA-based solutions are not immune to security concerns. This section explores the multifaceted of security implications associated with FPGA implementations, with a particular focus on the evolving landscape of post-quantum cryptography (PQC).

Vulnerabilities and threats in FPGA implementations

FPGAs have emerged as pivotal components in high-performance computing, embedded systems, and hardware acceleration. However, this increasing reliance on FPGA-based systems has exposed them to potential vulnerabilities and threats. Literature suggests that FPGA designs can be susceptible to side-channel attacks, configuration bitstream tampering, and data leakage [13–15]. Understanding these vulnerabilities is critical for devising robust FPGA security measures.

FPGA-based cryptographic solutions

FPGAs play a pivotal role in cryptographic applications, offering hardware acceleration for encryption and decryption tasks. This subsection reviews the literature on FPGA-based cryptographic solutions, including implementations of well-established algorithms such as Advanced Encryption Standard (AES) and Rivest Cipher (RC4) [16,17]. It also examines the performance gains achieved by utilizing FPGAs in cryptography, reinforcing their significance in secure data processing.

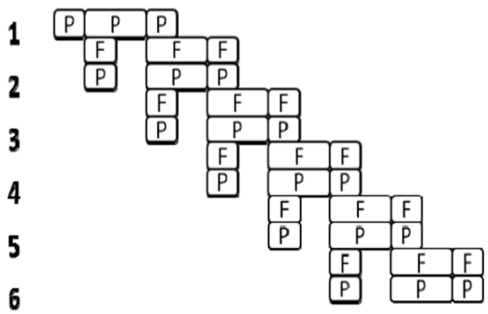

Fig. 7. Pipeline structure.

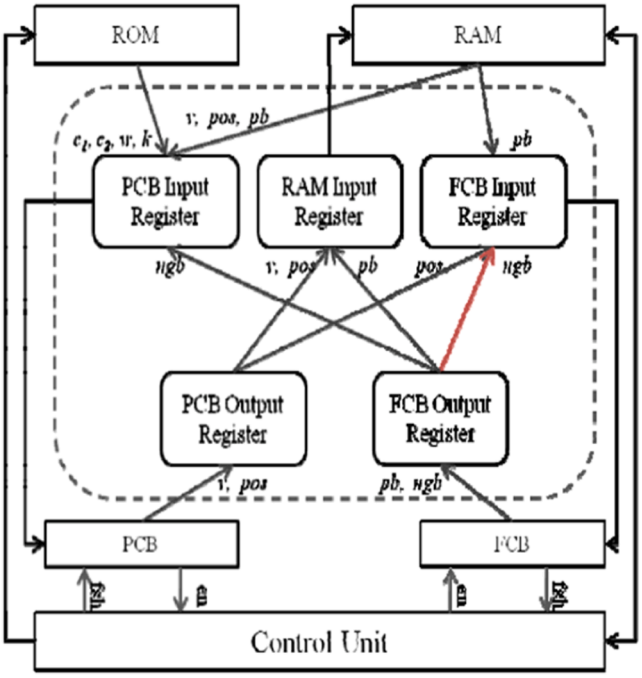

Fig. 8. Data path between register.

The advent of post-quantum cryptography (PQC). The emergence of quantum computing technology poses a profound challenge to conventional cryptographic methods, such as Elliptic Curve Cryptography (ECC) and RSA. This subsection delves into the recent developments in post-quantum cryptography (PQC) and its role in mitigating the threats posed by quantum adversaries. PQC algorithms, such as NTRUEncrypt and Lattice-based cryptography, are gaining traction as quantum-resistant alternatives [18,19]. The potential ramifications of PQC for FPGA-based security solutions are examined in light of the impending transition from classical to quantum-safe cryptographic practices.

The growing significance of Post-Quantum Cryptography (PQC) and the need for fault detection in lightweight ciphers are essential considerations in the context of our study. With the rise of quantum computing and its potential to undermine classical cryptographic techniques like RSA and ECC, PQC has emerged as a critical paradigm shift [20–22]. PQC is particularly crucial for ensuring the long-term security of digital communications and data protection. Furthermore, in constrained environments like IoT devices and FPGA-based systems, lightweight ciphers are paramount. Lightweight ciphers are designed to offer robust security while minimizing computational and memory demands, making them ideal for resource-constrained platforms. As we explore FPGA-based systems in this paper, the synergy between PQC and lightweight ciphers takes centre stage in safeguarding future-proof security within such resource-limited settings.

To gain a comprehensive understanding of these topics, it is essential to delve into specific areas of research that provide valuable insights. The study on Curve448 and Ed448 on Cortex-M4 delve into the practical implementation of Post-Quantum Cryptography (PQC) algorithms, specifically focusing on Curve448 and Ed448, within the constraints of Cortex-M4 platforms [23]. This exploration illuminates the challenges and opportunities in adapting PQC for resource-limited devices. Ongoing research initiatives on SIKE on Cortex-M4 are dedicated to optimizing the SIKE (Supersingular Isogeny Key Encapsulation) algorithm for Cortex-M4 platforms, ensuring its suitability for resource-constrained environments [24,25]. These efforts seek to address the unique requirements and performance considerations of lightweight cryptographic solutions. Recent developments in the third round of the SIKE competition place a strong emphasis on fine-tuning

SIKE for ARM Cortex-M4 platforms. This research is instrumental in highlighting the adaptability and utility of SIKE in constrained settings, further fortifying its potential [26]. Researchers have undertaken the task of implementing PQC algorithms like Kyber on the more resource-endowed 64-bit ARM Cortex-A architectures. Their work addresses not only the performance aspects but also the critical post-quantum security considerations inherent to these platforms [27].

Fault detection in lightweight ciphers. In the context of securing FPGA-based systems, fault detection in lightweight ciphers is crucial to ensure robust and reliable performance, especially in the presence of potential hardware faults. The integration of cryptographic accelerators into systems employing Ed25519, a widely adopted digital signature algorithm, is an area of exploration aimed at enhancing cryptographic performance [28]. In this work, the significance of optimizing cryptographic functions within resource-constrained environments. Investigating the Supersingular Isogeny Diffie–Hellman key exchange on 64-bit ARM platforms provides insights into its compatibility and performance within resource-rich architectures. This research is particularly relevant for applications demanding heightened security [29,30].

These research topics form the foundation for our understanding of Post-Quantum Cryptography and fault detection in lightweight ciphers. They provide valuable insights into the ongoing efforts to address security concerns and optimize cryptographic algorithms for various hardware platforms, including Cortex-M4, ARM Cortex-A, and FPGA-based systems.

Bridging FPGA implementations with PQC. To secure FPGA-based systems against quantum threats and maintain the integrity of sensitive data, it is imperative to bridge the gap between FPGA implementations and post-quantum cryptography. This section investigates the interplay between FPGA technology and PQC. It explores how FPGA architectures can be harnessed to accelerate PQC algorithms while ensuring their resilience against quantum attacks [31,32].

Lightweight cryptography (LWC) in FPGA implementations. In recent years, the field of lightweight cryptography (LWC) has gained prominence as a specialized area within the broader domain of cryptographic research. LWC is characterized by its fundamental goal of designing cryptographic primitives and protocols that are well-suited for resource-constrained environments, such as embedded systems, Internet of Things (IoT) devices, and notably, FPGA-based implementations. As the demand for efficient and secure processing in FPGA applications continues to grow, the principles of LWC hold particular relevance [33,34].

One of the central tenets of LWC is the optimization of computational resources. FPGA devices, known for their parallel processing capabilities, often operate under resource constraints, making resource-efficient cryptographic building blocks a valuable asset. By incorporating lightweight cryptographic primitives, FPGA implementations can strike a balance between high-performance computing and efficient resource utilization. Security remains paramount in FPGA-based solutions, especially when handling sensitive data or communications. LWC algorithms are designed to provide robust security even in scenarios where computational resources are limited. The ability to maintain security without imposing excessive resource overhead aligns seamlessly with the requirements of FPGA implementations in various applications [35,36].

FPGA devices excel in parallel processing, a feature that is highly conducive to many metaheuristic optimization algorithms. Lightweight cryptographic algorithms, characterized by their computational and memory efficiency, complement FPGA architectures well. This synergy empowers FPGA-based implementations to harness parallelism effectively, resulting in both high-performance and secure solutions. FPGA-based systems often operate in environments where data privacy and confidentiality are paramount. Lightweight cryptography, with its focus on protecting data with minimal computational burden, serves as a natural choice for ensuring the privacy and confidentiality of sensitive information within FPGA applications [37,38].

In light of these considerations, the incorporation of lightweight cryptographic building blocks emerges as a pertinent aspect in the context of FPGA implementation of metaheuristic optimization algorithms. The synergy between FPGA capabilities, the principles of LWC, and the demand for secure, resource-efficient solutions underscores the relevance of exploring lightweight cryptography as a valuable tool in optimizing FPGA-based metaheuristic implementations.

Post-quantum cryptography (PQC) and lightweight ciphers in resource-constrained environments

Post-Quantum Cryptography (PQC) has gained immense importance due to the impending threat posed by quantum computing to traditional cryptographic methods [20,39]. Quantum computers have the potential to break widely used encryption techniques, such as RSA and ECC, by leveraging their superior computing power. This creates a pressing need to transition to cryptographic methods that are resistant to quantum attacks.

PQC represents a paradigm shift in cryptographic research. It explores alternative mathematical foundations and encryption techniques that can withstand quantum attacks [40]. The importance of PQC lies in its ability to ensure the long-term security of digital communications, data protection, and privacy [31]. In constrained environments, such as IoT devices and resource-limited systems, PQC becomes even more critical because it offers a way to secure communications without imposing excessive computational overhead [41].

Importance of lightweight ciphers in constrained environments. Lightweight ciphers play a crucial role in constrained environments, where resources such as power, memory, and processing capabilities are limited. These environments include IoT devices, embedded systems, and even FPGA implementations. Lightweight ciphers are specifically designed to offer robust security while minimizing computational and memory requirements [42]. The importance of lightweight ciphers in constrained environments can't be overstated. They enable secure communication and data protection without burdening the already limited resources of these devices. Their efficiency and suitability for low-power, resource-constrained hardware make them a natural choice in scenarios where traditional ciphers might be impractical [43].

FPGA-based systems with post-quantum cryptography (PQC) and lightweight ciphers. The increasing reliance on FPGA-based systems, such as the one we explore in this paper, has ushered in tremendous opportunities. However, with these opportunities come challenges related to security, especially in a world where the threat of quantum computing looms large. The advent of quantum computing poses a profound challenge to conventional cryptographic methods, such as Elliptic Curve Cryptography (ECC) and RSA. To address this challenge and to secure FPGA-based systems against quantum threats, there is a growing need to integrate Post-Quantum Cryptography (PQC) and lightweight ciphers.

Post-Quantum Cryptography (PQC) represents a critical paradigm shift in the field of cryptography. It offers cryptographic solutions that can withstand the computational power of quantum adversaries. Recent developments in PQC, including algorithms like NTRUEncrypt and Lattice-based cryptography [18,19], have brought quantum-resistant alternatives to the forefront. Similarly, Lightweight Ciphers (LWC) have gained prominence as specialized cryptographic tools designed for resource-constrained environments, which includes FPGA-based systems. These ciphers optimize computational resources without compromising security, making them particularly relevant in this context [33,34]. This paper explores the synergy between FPGA capabilities, the principles of LWC, and the demand for secure, resource-efficient solutions. It highlights how FPGA-based systems,

operating in environments where data privacy and confidentiality are paramount, can benefit from the incorporation of lightweight cryptographic building blocks.

As we delve into the FPGA implementation of metaheuristic optimization algorithms, the role of PQC and lightweight ciphers takes center stage in ensuring future-proof security in resource-constrained settings.

Error detection and fault diagnosis mechanisms in lightweight ciphers

The successful deployment of lightweight ciphers in resource-constrained environments, such as FPGA-based systems, hinges on their ability to maintain correct operation and security even in the presence of errors or faults. This section delves into the critical aspect of error detection and fault diagnosis mechanisms within lightweight ciphers, shedding light on their significance and the relevant research in the field.

Error detection in lightweight Welch-Gong (WG)-oriented streamcipher WAGE. The lightweight cipher WAGE, which draws from the Welch-Gong (WG) design strategy, has gained attention for its efficiency and security characteristics. Error detection mechanisms play a pivotal role in ensuring the proper functioning of ciphers like WAGE, particularly in resource-constrained environments. Research endeavours have explored these mechanisms, emphasizing the importance of error detection to maintain the integrity of lightweight ciphers [44].

Error detection reliable architectures of Camellia block cipher. The Camellia block cipher, a widely recognized cryptographic algorithm, requires robust fault detection strategies to safeguard its security and reliability. Recent work has addressed the development of reliable architectures and techniques for error detection and correction within the context of the Camellia block cipher. These advancements are essential for ensuring the resilience of cryptographic systems to faults [45,46].

Fault diagnosis of low-energy Midori cipher. The Midori cipher, designed with a focus on energy-efficient applications, demands fault diagnosis mechanisms to maintain its security in low-power environments. Recent research has explored fault diagnosis techniques specifically tailored to the energy-efficient Midori cipher. The incorporation of such mechanisms becomes crucial to ensure fault tolerance in cryptographic systems with low energy consumption [47,48].

Block cipher QARMA with error detection mechanisms. QARMA, a block cipher designed for lightweight and efficient applications, benefits from the integration of error detection mechanisms to enhance its fault tolerance. Relevant studies have emphasized the importance of incorporating these mechanisms within the QARMA block cipher, thereby enhancing its resilience and reliability [49].

Incorporating error detection and fault diagnosis mechanisms is vital to the successful implementation of lightweight ciphers in constrained environments. These mechanisms bolster the security and reliability of cryptographic systems, ensuring their continued operation and data integrity. The research presented in these references underlines the importance of addressing these aspects in the context of lightweight ciphers and provides valuable insights for their efficient and fault-tolerant utilization.

Hardware/software platform selection and security implications in FPGA implementation

The choice of hardware/software platforms plays a pivotal role in the security implications of FPGA implementation. FPGA (Field-Programmable Gate Arrays), ASIC (Application-Specific Integrated Circuits), ARM (Advanced RISC Machine), and RISC-V (Reduced Instruction Set Computer - V) each present distinct advantages and considerations when it comes to security in the context of your metaheuristic optimization algorithm.

FPGAs offer flexibility and high processing speed, making them an attractive choice for various applications. However, this comes with a trade off in the context of security. The reconfigurable nature of FPGAs allows for quick adaptations but also introduces vulnerabilities if not properly secured. It's crucial to consider FPGA bitstream security to safeguard against unauthorized access or tampering. Moreover, as FPGAs are often used in constrained environments, ensuring security without significant computational overhead is vital.

ASICs, on the other hand, provide dedicated, efficient processing tailored to specific tasks, enhancing security in some aspects. Since ASICs are not reconfigurable, they can be more resistant to certain attacks, but they are less adaptable for algorithm updates or changes. In the context of your metaheuristic optimization algorithm, ASICs might provide a balance between security and performance, particularly when considering fixed, long-term deployment.

ARM and RISC-V architectures often serve as the foundation for many embedded systems, including FPGA-based implementations. Security considerations extend to both the choice of processor architecture and the implementation of secure boot processes, cryptographic accelerators, and secure coding practices. RISC-V, being open-source, allows for more transparency and customization, but security must be meticulously managed.

The security of various hardware and software platforms like FPGA, ASIC, ARM, and RISC-V relies on different aspects, which include looking at secure development methods to prevent vulnerabilities, the use of specialized cryptographic components, secure boot procedures to ensure trusted software execution, and the balance between performance and adaptability in the context of security. These considerations play a critical role in hardware/software platform selection. This is to ensure the security of the FPGA implementation of metaheuristic optimization algorithms, particularly in diverse application domains with varying security requirements.

Methodology

Simulated Kalman filter

Through the process of elimination, the metaheuristic chosen for FPGA implementation was the Simulated Kalman Filter (SKF). The bio-inspired metaheuristics Barnacles Mating Optimizer and Orca Predation Algorithm proposed steps that were too complex and numerous. This would result in a difficult implementation of the algorithm as it may incur high costs to implement all the different states and modules of the algorithm. The Single-Agent Finite Impulse Response Optimizer proposes an iterative process with a sub-iterative process which introduces the same problem as mentioned above.

The SKF metaheuristic is composed of 4 major components namely the population generation, fitness evaluation of agents, fitness evaluation comparison and storage, and the Kalman Filter components which are the predict, measure, and estimate steps as shown in Fig. 4. These steps are carried out iteratively until a stopping condition is met which is typically the maximum number of iterations. There are several variables of the Kalman Filter components which are significant such as the initial error covariance estimate, P(0), process noise, Q, measurement noise, R, and the Kalman gain, K.

Before the first iteration starts, several parameters need to be defined such as the number of agents, N, number of dimensions, D, and maximum number of iterations, t_{max}. Then the initial generation of population is done by generating a random floating-point value between the range of −100 to 100 and loading it into a multidimensional array X which has N columns and D rows.

Then the evaluation step is carried out by evaluating the fitness of the agents through the activation function. The activation function of the original SKF metaheuristic utilizes the CEC2014 benchmark function

which contains a set of 30 activation functions. Table 1. illustrates the generation of population of 3 agents with 3 dimensions and fitness evaluation using the sphere activation function. The sphere activation function is described in Eq. (1).

$$Fitness(t) = \sum_{n=1}^{D} D(t)^2 \tag{1}$$

After all agents have their fitness evaluated, all the fitness evaluation values are compared and the agent with the best fitness evaluation for the current iteration is selected as the $X_{best}(t)$. The selection process for $X_{best}(t)$ depends on whether it is a minimization problem or maximization problem. Eq. (2) is used to determine $X_{best}(t)$ if it is a minimization problem and Eq. (3) is used if it is a maximization problem. The $X_{best}(t)$ is then used to update the best-so-far solution of the run, X_{true} by replacing the best-so-far solution with the $X_{best}(t)$ value if $X_{best}(t)) < X_{true}$ for minimization problems and $X_{best}(t)) > X_{true}$ for maximization problems.

$$X_{best} = min_{i\in 1,\ldots,n} fitness_i(X(t)) \tag{2}$$

$$X_{best} = max_{i\in 1,\ldots,n} fitness_i(X(t)) \tag{3}$$

In the first iteration, the predict step is carried out by setting the initial error covariance to 1000, P(0) = 1000, the process noise is set to 0.5, $Q = 0.5$, and the measurement noise is set to 0.5, $R = 0.5$. Then the error covariance is calculated using Eq. (4).

$$P = P + Q \tag{4}$$

Then the measurement step is carried out using Eq. (5). where the X is the position of the agents and Y is the measured value of the agents.

$$Y = X + (sin(rand(n, 1) \times 2\pi)) \times |X - X\_true| \tag{5}$$

Then the estimate step is proceeded to calculate the Kalman gain value as shown in Eq. (6). Then the position of agents is updated using the Eq. (7) and lastly the new error covariance is updated through the Eq. (8).

$$K = \frac{P}{(P + R)} \tag{6}$$

$$X = X + K \times (Y - X) \tag{7}$$

$$P = (1 - K) \times P \tag{8}$$

At the end of the estimate step of the Kalman Filter components, the stopping condition is checked and the run stops if the maximum number of iterations have been achieved. Else the run continues on to the next iteration and the steps are repeated from the evaluation of fitness agent.

Discrete simulated Kalman filter adaptation

The original SKF metaheuristic was simulated using MATLAB utilizing floating-point values. The operation of this simulation is represented in Table 2 where the agent positions as well as its fitness evaluation are represented by whole numbers, decimal point, and its corresponding fractional part. This introduced complications into the design of the FPGA implementation as it is difficult to represent the fractional parts of a number in a digital system. Hence, the original SKF was adapted into a Discrete SKF to remove this complication.

Table 1
Generation of population and fitness evaluation for SKF.

ArrayX		Agents		
		X1	*X2*	*X3*
Dimension	D1	−10.6638	14.53027	−10.9522
	D2	48.57666	−20.4318	40.2808
	D3	53.30432	−56.9214	11.46109
Evaluation		5314.759	3868.633	1873.85

Table 2
Generation of population and fitness evaluation for Discrete SKF.

Array X		Agents		
		X1	*X2*	*X3*
Dimension	D1	−11	15	−11
	D2	49	−20	40
	D3	53	−57	11
Evaluation		5331	3874	1842

The Discrete SKF utilizes the same minimization sphere function but utilizes only integer values. This is done by rounding all the floating-point agent position values to the nearest integer. This would consequently produce fitness evaluation values in whole number as well. The generation of population and fitness evaluation by Discrete SKF is illustrated in Table 2.

The Kalman gain which was previously calculated using Eq. (6) is set to a fixed value of $K = 0.5$. This is done because a multiplication of 0.5 can be translated into a division by 2 which can be operated in binary as a logical right shift of 1 bit. The deliberate value of 0.5 is set after thorough investigation of the transition of the Kalman gain in the original SKF MATLAB simulation. The Kalman gain can be observed to slowly decrease from 0.9995 at the first iteration and converges to 0.6180 by 7th iteration as shown in Table 3.

Table 4 presents a detailed account of the population generation and fitness evaluation within the context of the Binary Simulated Kalman Filter (SKF) implementation. The population is composed of individual agents denoted as X1, X2, and X3, each possessing multiple dimensions represented by D1, D2, and D3. These dimensions encode binary sequences that encapsulate specific parameters or characteristics pertinent to the optimization algorithm. For instance, in D1, Agent X1 is associated with the binary sequence ``11,110,101,'' while Agent X2 exhibits ``00,001,111,'' and Agent X3 features ``11,110,101.'' The fitness evaluation, as depicted in the ``Evaluation'' row, comprises binary bits signifying fitness scores attributed to each agent. These fitness scores are indicative of the performance of each agent concerning the optimization problem under consideration. The table serves as an informative visual representation of the binary population, dimensions, and fitness assessments, offering insights into the algorithm's functionality and its capacity to adapt to the optimization task. These results constitute a pivotal component of the iterative optimization process facilitated by the Binary SKF algorithm, guiding its pursuit of optimal solutions within the population.

This Discrete SKF was then simulated in MATLAB to verify its results. The significance of the adaptation from the original to discrete version of SKF is that the discrete version can be readily implemented in a digital system by representing the integer values as binary values. The implementation of Discrete SKF as Binary SKF is illustrated in 00 where the integer values are represented in 2 s complement binary values.

Binary simulated Kalman filter behavioral modeling

Each section of the Binary SKF metaheuristic was segmented and

Table 3
Progression of Kalman gain in original SKF.

Iteration, t	P	Q	R	K
1	1000	0.5	0.5	0.9995
2	0.4998	0.5	0.5	0.6666
3	0.3333	0.5	0.5	0.6250
4	0.3125	0.5	0.5	0.6190
5	0.3095	0.5	0.5	0.6182
6	0.3091	0.5	0.5	0.6181
7	0.3090	0.5	0.5	0.6180
8	0.3090	0.5	0.5	0.6180
9	0.3090	0.5	0.5	0.6180

Table 4
Generation of population and fitness evaluation of Binary SKF.

Array X		Agents		
		X1	X2	X3
Dimension	D1	11110101	00001111	11110101
	D2	00110001	11101100	00101000
	D3	00110101	11000111	00001011
Evaluation		0001010011010011	0000111100100010	0000011100110010

individually designed into separate modules using System Verilog. The major components that needed to be designed were the random number generator to generate the initial population of agents, Random-Access Memory (RAM) to store the agent position and measurement values, Activation Function module to evaluate the fitness of each agent according to Eq. (1), Measure module to carry out measurement calculations as shown in Eq. (5), and the Estimate module to update the position of agents according to Eq. (7). The initial design produced is capable of processing 10 dimensions. This design is repurposed to produce 3 different variants which are the 5-dimension, 10-dimension and 20-dimension Binary SKF for FPGA implementation.

Behavioral modeling of modules

The Random Number Generator (RNG) module is designed upon the concept of Linear Feedback Shift Register to produce a sequence of random numbers which are dependent on the initial seed provided. The generated values are in 8-bits which represents the initial position of agents. The values are internally bounded within −100 to 100 to ensure the values produced are in the search region. The Random-Access Memory (RAM) is designed to temporarily store the position values and measured values of all the dimensions of the agents. This module is essentially a multidimensional array of registers with N number of columns and D number of rows. RAM_X is instantiated to store the position values of all the dimensions of the agents generated by the RNG module and the Estimate module. Meanwhile RAM_Y is instantiated to store the output of the Measure module.

The activation function module facilitates the computation required to evaluate the fitness of agents as described in Eq. (1). The module receives input from RAM_X which holds the position values of all the agents. The module then determines the agent with the overall best evaluation of the current iteration and stores it at the Xbest register. The measure module facilitates the calculations required to carry out the measure step as described in Eq. (5). It receives inputs from RAM_X which holds the current position values of all agents and best_agent_D# registers which stores the position value of the Xtrue register. This module then offloads the calculated values into RAM_Y. This module

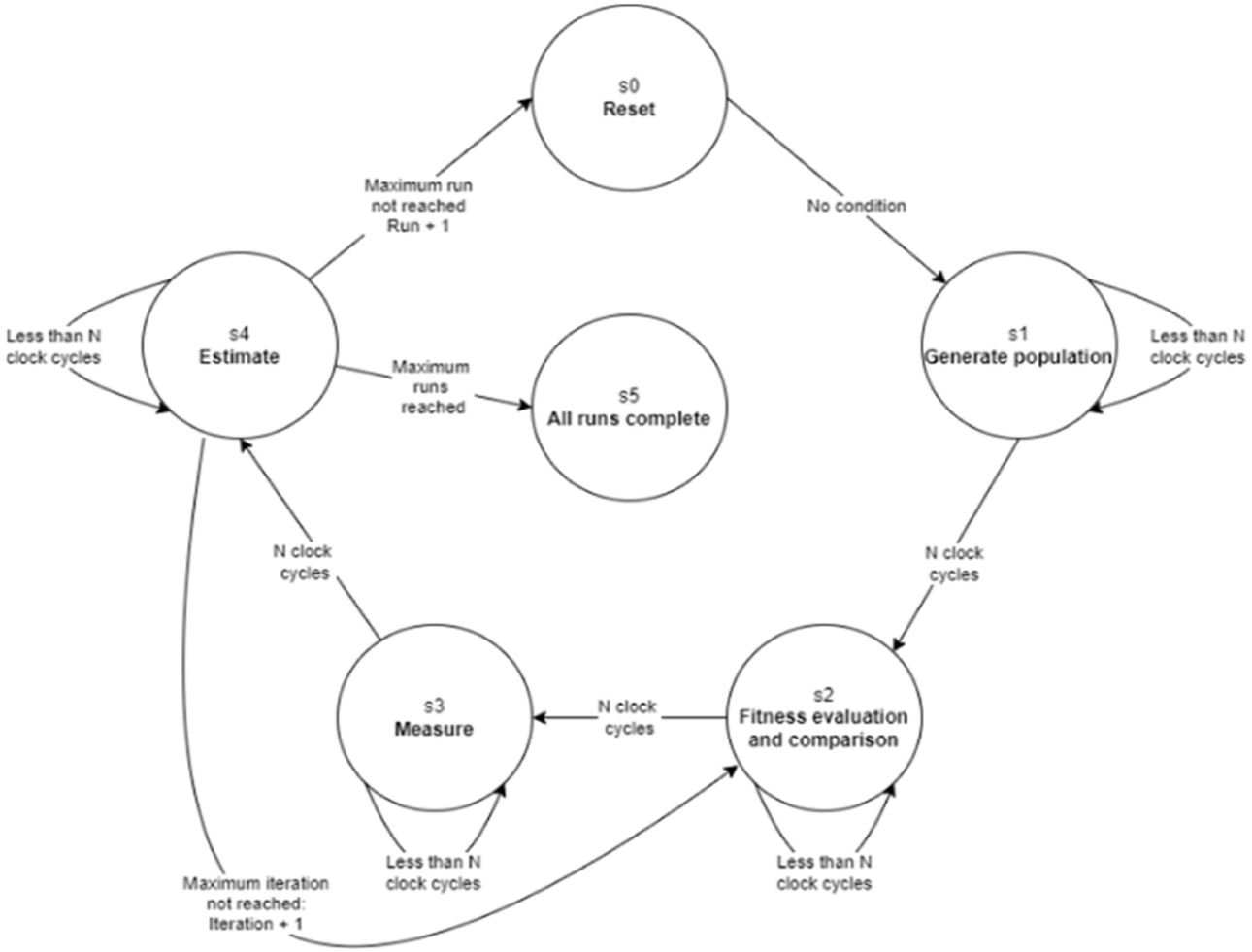

Fig. 9. Modules generated through behavioral modeling.

facilitates the final step of SKF by computing Eq. (7). The module receives input from RAM_X containing current position values of all agents and RAM_Y containing measured values from the Measure module. The module then updates the current position values of all agents and overwrites RAM_X with the new agent position values for the next iteration.

Parallel-in-parallel-out configuration of modules

During the behavioral modeling of the required modules, it was discovered that it was possible to produce modules with multiple input and output ports. This was significant as it enabled a module to receive multiple inputs, process the inputs, and produce multiple outputs at once. This Parallel-In-Parallel-Out (PIPO) configuration was exploited to maximize the number of inputs and outputs of each module and consequently improve their performance. The number of ports is dependent on the number of dimensions for that implementation. The modules designed are shown in Fig. 9.

Finite state machine controller

To ensure proper timing and flow of data between modules, a finite state machine (FSM) was modeled using System Verilog. The FSM instantiates all the necessary modules and the necessary wires for interconnection between modules. The FSM has 6 states in sequence of s0 Reset, s1 Generate population, s2 Fitness evaluation and comparison, s3 Measure, s4 Estimate, and s5 Complete. A simplified state diagram of the FSM is illustrated in Fig. 10.

Hardware components

The board chosen for the implementation of the Binary SKF was the DE10 - Lite which is powered by the 10M50DAF474G chip produced by Altera. Included on the board is a clock generation chip which outputs a stable 50 MHz clock cycle to drive the FPGA. The FPGA chip is sufficiently powerful to power the design as it contains 50 thousand logical elements, and 144 units of 18 × 18 multiplier.

Meanwhile, the MATLAB simulation of Discrete SKF is ran on a HP ProBook 440 G7 which is equipped with a 10th generation Intel processor namely the Intel Core i5 – 10210U CPU and is paired with 16 gigabytes of RAM.

Results and discussion

The run parameter set for both the MATLAB simulation and FPGA implementation is set to 50 maximum runs, 5000 maximum iterations, 50 agents, and fixed Kalman gain of 0.5. The runs are conducted 3 times on MATLAB using 3 different dimension values which are 5-dimensions,

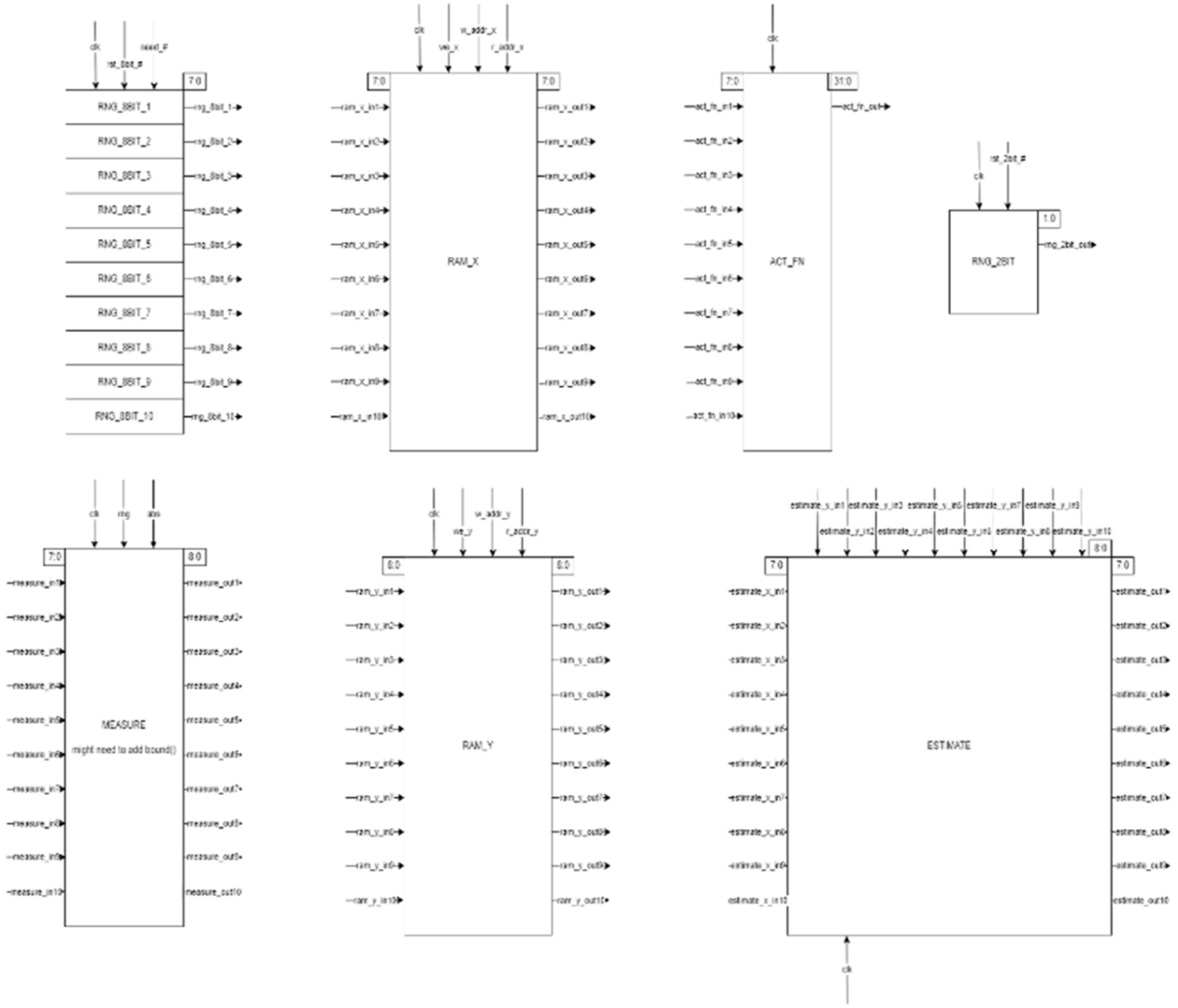

Fig. 10. State diagram of finite state machine controller.

followed by 10-dimensions, and lastly 20-dimensions. The FPGA is programmed with 3 different designs and ran 3 separate times to facilitate the run of the 3 different variants namely the 5-dimension, 10-dimension, and 20-dimension. The time taken to complete all calculations, the value of Xtrue and chip utilization of each design is noted is compiled into Table 5.

Chip utilization

The logical elements utilization increases by 43 % when the number of dimensions increases from 5 to 10. Meanwhile, an increment of 46 % is observed when the number of dimensions increases from 10 to 20. The utilization of logical elements on the FPGA is a crucial aspect of our implementation. We configured our FPGA with a fixed Kalman gain of 0.5 and conducted multiple runs with different dimensions (5, 10, and 20). The logical elements utilization increased by 43 % when the number of dimensions increased from 5 to 10. Moreover, an increment of 46 % was observed when the number of dimensions increased from 10 to 20. The increase in logical elements utilization with the rise in dimensions highlights the resource-intensive nature of our FPGA implementation. This finding highlighted the importance of optimizing FPGA resources efficiently to accommodate higher dimensions without exceeding logical element capacity.

Processing speed

The time taken to complete the simulation on MATLAB increases by 25 % when the number of dimensions increased from 5 to 10. Meanwhile increasing the dimensions from 10 to 20 further increased the time taken by 33 %. This shows that the time taken to complete the simulation increases as the number of dimensions increases for the MATLAB simulation. However, the FPGA implementation completes all its runs in 0.780 s. It is observed that the time taken to complete the run remains the same despite increasing the number of dimensions for the FPGA implementation. This is because the modules of the FPGA implementation are designed using the PIPO configuration which enables all dimensions of a single agent to be loaded and processed in a single clock cycle. Consequently, the PIPO configuration of modules will result in the same processing time despite an increment or decrement of number of dimensions.

Accuracy of result

The accuracy of our results is crucial for evaluating the effectiveness of our FPGA implementation. We assessed the accuracy of both the MATLAB simulated Discrete SKF and the FPGA-implemented Binary SKF. Both the MATLAB simulated Discrete SKF and the FPGA-implemented Binary SKF exhibited challenges in producing accurate results. In the MATLAB simulation, the inaccuracy stemmed from rounding floating-point values to integers and the use of a fixed Kalman gain of 0.5. These factors reduced resolution and accuracy. The error introduced in the MATLAB simulation carried forward to the FPGA implementation, which uses binary format to represent integer values. The diminished accuracy of both simulations highlights the importance of fine-tuning our FPGA implementation to mitigate these challenges. The use of binary representation can lead to precision loss, particularly in cases where floating-point accuracy is essential. Future work should focus on optimizing the FPGA design to enhance result accuracy while maintaining the efficiency achieved through PIPO configuration.

Comparison with previous works

Comparing the implementation of Binary SKF to other works in the literature poses challenges due to variations in the employed metaheuristics, testing methodologies, and configurations. A comparative analysis of these unique approaches is provided in Table 6.

Binary SKF FPGA implementation speed improvements

This section further investigates the areas within the Binary Simulated Kalman Filter (Binary SKF) algorithm where speed improvements are significant. It focuses on operations that have been optimized through FPGA parallelism or other hardware-accelerated techniques, which are agent interaction, solution evaluation, control logic, data communication, resource allocation and iteration management. Within the Binary SKF algorithm, the interactions and information exchange among individual agents play a crucial role. FPGA implementations accelerate these interactions, especially when numerous agents are involved, leading to substantial speed improvements.

Binary SKF typically involves evaluating the quality of solutions generated by agents. FPGA hardware expedites this evaluation process, leading to faster convergence and more efficient optimization. The control and coordination of various modules in the Binary SKF algorithm benefited from FPGA acceleration. This includes the management of the optimization process, the decision to terminate, and the synchronization of operations. The FPGA's parallel processing capabilities enhanced the communication of data between modules. This acceleration is particularly valuable in multi-agent systems, where data exchange is frequent.

In applications where Binary SKF is used for resource allocation, such as energy management or manufacturing, the allocation process can experience significant speed improvements on FPGA hardware. The speed of iterating through the optimization process is substantially enhanced through FPGA implementation, reducing the time needed to find optimal solutions.

Project limitations

Accuracy and precision

In order to accurately represent the original SKF metaheuristic, the employment of floating-point arithmetic is vital. This is because the Kalman gain which is calculated in the estimation step of SKF is in the form of floating-point. In the implementation of Binary SKF, the Kalman gain is set to a fixed value of 0.5 which impedes the performance of the metaheuristic during the exploitation phase and consequently fails to find the optimum solution. By modifying each module to facilitate arithmetics in floating point, the implementation will be able to produce a dynamic Kalman gain and reap the benefits of the original SKF which boasts great performance in the exploitation process enabling it to produce accurate results [4].

Resource constraints

The implementation of floating-point arithmetic would significantly

Table 5
Generation of population and fitness evaluation for Binary SKF.

Array X		Agents		
		X1	*X2*	*X3*
Dimension	D1	11110101	00001111	11110101
	D2	00110001	11101100	00101000
	D3	00110101	11000111	00001011
Evaluation		0001010011010011	0000111100100010	0000011100110010

Table 6
Compiled result of simulation and FPGA implementation.

Platform	Agents, N	Dimensions, D	Xtrue	Expected value	Time taken (s)	Logical elements	Registers
Simulation	50	5	14	0	27.329	–	–
	50	10	503	0	36.481	–	–
	50	20	3936	0	54.571	–	–
FPGA	50	5	32	0	0.780	8220	4686
	50	10	253	0	0.780	14,637	9146
	50	20	4749	0	0.780	27,413	18,066

impact the resource consumption of the implementation. This is because each module and register would require modifications to be able to process calculations and store floating-point values adhering to the IEEE 754 format. The implementation of the 20-dimension Binary SKF consumed up to 55 % of the logical elements of the FPGA. The inclusion of floating-point arithmetic would further increase the logical elements required and may exceed the hardware limit [11].

Security considerations

The implemented Binary SKF can be vulnerable to side-channel attacks and data leakages. This is because no security features were implemented to prevent these occurrences. The implementation of LWC can be an effective and resource efficient method of intercepting side-channel attacks by encrypting the computation and transmission of sensitive data.

Scalability

The implementation of the PIPO configuration of models have shown to enable parallel processing of multidimensional optimization problems. This is because in a single clock cycle, every dimension of an agent is loaded into the subsequent modules and processed at once. This enables the implementation to process optimization problems with a large range of dimensions without affecting the time taken to complete. However, this speed comes at the cost of resource utilization as the increment in dimensions would consequently increase the number of logical elements consumed. This issue could be minimized by simplifying and modifying the modules to enable resource sharing.

Potential implications and applications of the Binary SKF algorithm in real-world optimization problems

The Binary Simulated Kalman Filter (Binary SKF) algorithm in real-world optimization problems. This algorithm exhibits versatility across various problem domains, offering potential benefits over existing optimization techniques. In the domain of supply chain management, Binary SKF can effectively address intricate demand forecasting, inventory management, and production planning challenges. In the financial sector, it is well-suited for real-time portfolio optimization, asset allocation, and risk management, courtesy of its rapid processing capabilities. Its adaptability extends to wireless sensor network configuration, optimizing sensor placement for maximal coverage and minimal energy consumption. In engineering design, Binary SKF aids in tasks such as parameter tuning, component placement, and circuit design, fostering efficiency and cost-effectiveness. Moreover, its application in healthcare for biomedical parameter estimation enhances the accuracy of physiological modeling.

Enhancing accuracy in Binary SKF implementation

Improving the precision of Binary Simulated Kalman Filter (SKF) implementation is of paramount importance, particularly while considering the constraints posed by the Discrete SKF and the fixed Kalman gain. To achieve this, various strategies have been explored in the context of conceptual simulations, and further possibilities exist for enhancing the precision of Binary Simulated Kalman Filter (SKF) implementation. One promising approach involves moving away from binary implementation and the fixed Kalman gain. Instead, it proposes replacing them with a novel version of the Discrete SKF, which leverages concepts like distance evaluation with state encoding [50]. This transition holds the potential to refine the SKF's precision and suitability for combinatorial optimization problems. A complementary strategy involves the development of Discrete SKF variants tailored to discrete search spaces. This adaptation necessitates modifying measurement and estimation methods within the SKF to align with the requirements of discrete search spaces [51]. Such modifications hold promise for enhanced precision in solving problems like the Traveling Salesman Problem.

Instead of adhering to the fixed 0.5 gain, an adaptive approach allows the algorithm to dynamically adjust the gain during runtime, responding to the optimization process's progress. This adaptability can lead to improved accuracy and convergence. Employing fixed-point arithmetic with adjustable scaling factors offers a path to fine-tune precision. This strategy aims to balance computational accuracy with resource efficiency. Combining the Binary SKF algorithm with complementary optimization methods, such as Genetic Algorithms or Particle Swarm Optimization, creates a hybrid approach. This integration allows leveraging the strengths of multiple algorithms to enhance precision and robustness. Post-processing techniques, like gradient-based optimization or simulated annealing, can be employed to fine-tune discrete solutions. Iteratively optimizing results through these methods can significantly improve overall accuracy. Exploring these strategies not only addresses existing limitations but also opens new avenues for advancing the precision of SKF implementations. The choice of strategy may depend on the specific problem domain and the trade-offs between precision, computation, and adaptability. These research directions contribute to the ongoing quest for efficient and accurate metaheuristic optimization solutions (Table 7).

Conclusion

The work provided a detailed overview of the SKF metaheuristic, breaking down its major components, from population generation to the Kalman Filter steps, highlighting the critical variables. The utilization of MATLAB simulations to verify the transition from the original SKF to a discrete version that can be implemented in digital systems was presented. The paper effectively discussed the behavioral modeling of modules, emphasizing the advantages of the Parallel-In-Parallel-Out (PIPO) configuration. The exploration of FPGA implementation underlined its significant speed improvements, particularly in scenarios involving agent interactions, solution evaluation, control logic, data communication, resource allocation, and iteration management. The observed benefits in processing speed, despite an increase in dimensions, were attributed to FPGA's parallel processing capabilities. The study, however, noted the challenges related to the accuracy and precision of results. It recognized that achieving greater accuracy would necessitate a transition from a fixed Kalman gain to dynamic adaptation and the use of floating-point arithmetic. These changes would increase resource consumption, potentially exceeding hardware limits and necessitating optimization. Moreover, security considerations were raised, emphasizing the vulnerability of the implementation to side-channel attacks. The practical applications of Binary SKF in various domains, such as supply chain management, finance, wireless sensor networks, and

Table 7
Binary SKF FPGA Implementation Comparisons.

Implementation	Parallel implementation	Pipeline architecture	Floating point arithmetic	PIPO configuration
Metaheuristic used	Particle swarm optimization			Binary Kalman filter
Metaheuristic configuration	Dimension: 3, 6, & 10 Particles: 5, 10, & 15 Max iterations: - Max runs: -	Dimension: 1 Particles: 10 Max iterations: 50 Max runs: -	Dimension: 2 Particles: 5, 10, & 20 Max iterations: 6000 Max runs: -	Dimension: 5, 10, & 20 Particles: 50 Max iterations: 5000 Max runs: 50
Activation Function used	Rastrigin	Non-standard function	Sphere, Rosenbrock, & Rastrigin	Sphere
Method	Implementation of multiple Particle Modules to calculate the fitness of multiple particles simultaneously	Implementation of a pipeline architecture by simplifying and restructuring the FSM	Introduction of floating point arithmetic module to improve computation accuracy and speed	Implementation of parallel input and output for each module to enable parallel computing of multiple dimensions
Result	Calculation speed improvements compared to previous implementations: 6 dimensions: 212×	Calculation speed improvement by 18 % compared to previous implementation	Average calculation speed improvement compared to previous implementations: Sphere: 281× Rastrigin: 46× Rosenbrock 30×	Calculation speed improvement compared to Matlab simulation: 5 dimension: 35.04× 10 dimension: 46.77× 20 dimension: 69.96×

engineering design, underlining its versatility. Strategies for enhancing accuracy and precision are discussed, suggesting the exploration of new implementations, dynamic gain adjustment, fixed-point arithmetic, hybridization with complementary optimization methods, and post-processing techniques. These strategies aimed to strike a balance between precision and efficiency in the context of specific problem domains. In conclusion, a comprehensive exploration of Binary SKF, from its FPGA implementation to strategies for improving its precision and accuracy. It opens new avenues for research and optimization in the field of metaheuristic algorithms.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We are sincerely grateful to Ministry of Higher Education Malaysia for the generous support through the Fundamental Research Grant Scheme - FRGS/1/2018/TK04/UMP/03/3. This has enabled the successful realization of our research at the UMP STEM Lab, Universiti Malaysia Pahang Al-Sultan Abdullah. We greatly appreciate the confidence and belief invested in our work, and we are committed to upholding the highest standards of academic rigor and excellence.

References

[1] T. Ab Rahman, Z. Ibrahim, N.A.A. Aziz, S. Zhao, N.H.A. Aziz, Single-agent finite impulse response optimizer for numerical optimization problems, IEEE Access 6 (2018) 9358–9374.
[2] Z. Ibrahim, N.H.A. Aziz, N.A.A. Aziz, S. Razali, M.S. Mohamad, Simulated Kalman filter: a novel estimation-based metaheuristic optimization algorithm, Adv. Sci. Lett. 22 (2016) 2941–2946.
[3] N.H. Abdul Aziz, Z. Ibrahim, N.A. Ab Aziz, M.S. Mohamad, J. Watada, Single-solution simulated Kalman filter algorithm for global optimisation problems, Sādhanā 43 (2018) 1–15.
[4] M.H. Sulaiman, Z. Mustaffa, M.M. Saari, H. Daniyal, Barnacles mating optimizer: a new bio-inspired algorithm for solving engineering optimization problems, Eng. Appl. Artif. Intell. 87 (2020), 103330.
[5] Y. Jiang, Q. Wu, S. Zhu, L. Zhang, Orca predation algorithm: a novel bio-inspired algorithm for global optimization problems, Expert Syst. Appl. 188 (2022), 116026.
[6] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, 1995, pp. 1942–1948.
[7] M.A. Khanesar, M. Teshnehlab, M.A. Shoorehdeli, A novel binary particle swarm optimization, in: 2007 Mediterranean Conference on Control & Automation, 2007, pp. 1–6.
[8] M. Kong, P. Tian, Introducing a binary ant colony optimization, in: Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4–7, 2006. Proceedings 5, 2006, pp. 444–451.
[9] M.A. Tawhid, K.B. Dsouza, Hybrid binary bat enhanced particle swarm optimization algorithm for solving feature selection problems, Appl. Comput. Inform. 16 (2018) 117–136.
[10] A.L. Da Costa, C.A. Silva, M.F. Torquato, M.A. Fernandes, Parallel implementation of particle swarm optimization on FPGA, IEEE Trans. Circuits Syst. II 66 (2019) 1875–1879.
[11] X. Cai, S. Ngah, H. Zhu, Y. Tanabe, T. Baba, Pipeline architecture of particle swarm optimization, in: 2010 IEEE/ACIS 9th International Conference on Computer and Information Science, 2010, pp. 3–8.
[12] A. Rathod, R. Thakker, FPGA realization of particle swarm optimization algorithm using floating point arithmetic, in: 2014 International Conference on High Performance Computing and Applications (ICHPCA), 2014, pp. 1–6.
[13] J.Y. Koh, T. Nandha Kumar, Review of side channel attacks and countermeasures of FPGA based systems, in: 2021 IEEE 19th Student Conference on Research and Development (SCOReD), 2021, pp. 102–107.
[14] S. Sunkavilli, Z. Zhang, Q. Yu, New security threats on FPGAs: from FPGA design tools perspective, in: 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2021, pp. 278–283.
[15] B. Erbagci, "Hardware-Entangled Inherently Secure Field Programmable Gate Arrays," 2018.
[16] A.J. Elbirt, W. Yip, B. Chetwynd, C. Paar, An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 9 (2001) 545–557.
[17] L. Bahadur and M.B. Badoniya, "Study of AES implementation on FPGA by using Rijndael Algorithm," 2014.
[18] A.K. Sherdel, B. Schneider, Post-Quantum Cryptography: An Introductory Overview and Implementation Challenges of Quantum-Resistant Algorithms, in: K. Hinkelmann, A. Gerber (Eds.), Proceedings of the Society 5.0 Conference 2022 - Integrating Digital World and Real World to Resolve Challenges in Business and Society 84, EasyChair, 2022, pp. 61–71. 10.29007/2tpw.
[19] E. Zeydan, Y. Turk, B. Aksoy, S.B. Ozturk, Recent advances in post-quantum cryptography for networks: a survey, in: 2022 Seventh International Conference On Mobile And Secure Services (MobiSecServ), 2022, pp. 1–8.
[20] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, et al., *Report on post-quantum cryptography* vol. 12: US Department of Commerce, National Institute of Standards and Technology …, 2016.
[21] D.J. Bernstein, Introduction to post-quantum cryptography, in: D.J. Bernstein, J. Buchmann, E. Dahmen (Eds.), Post-Quantum Cryptography, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 1–14.
[22] J.B.J. Ding, "Post-quantum cryptography," 2008.
[23] M. Hamburg, Ed448-Goldilocks, a new elliptic curveCryptol, IACR Cryptol. ePrint Arch. 2015 (2015) 625.
[24] D. Jao, L. De Feo, Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies, in: Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29–December 2, 2011. Proceedings 4, 2011, pp. 19–34.
[25] H. Seo, A. Jalali, R. Azarderakhsh, SIKE round 2 speed record on ARM Cortex-M4, in: Cryptology and Network Security: 18th International Conference, CANS 2019, Fuzhou, China, October 25–27, 2019, Proceedings 18, 2019, pp. 39–60.
[26] M. Anastasova, R. Azarderakhsh, M.M. Kermani, Fast strategies for the implementation of SIKE round 3 on ARM Cortex-M4, IEEE Trans. Circuits Syst. I 68 (2021) 4129–4141.

[27] P. Sanal, E. Karagoz, H. Seo, R. Azarderakhsh, M. Mozaffari-Kermani, Kyber on ARM64: compact implementations of Kyber on 64-bit ARM Cortex-A processors, in: International Conference on Security and Privacy in Communication Systems, 2021, pp. 424–440.
[28] D.J. Bernstein, N. Duif, T. Lange, P. Schwabe, B.-Y. Yang, High-speed high-security signatures, J. Cryptogr. Eng. 2 (2012) 77–89.
[29] A. Jalali, R. Azarderakhsh, M.M. Kermani, D. Jao, Supersingular isogeny Diffie–Hellman key exchange on 64-bit ARM, IEEE Trans. Dependable Secure Comput. 16 (2017) 902–912.
[30] H. Seo, P. Sanal, R. Azarderakhsh, SIKE in 32-bit ARM processors based on redundant number system for NIST level-II, ACM Trans. Embed. Comput. Syst. (TECS) 20 (2021) 1–23.
[31] H. Li, Y. Tang, Z. Que, J. Zhang, FPGA accelerated post-quantum cryptography, IEEE Trans. Nanotechnol. 21 (2022) 685–691.
[32] Z. Ni, A. Khalid, M. O'Neill, High performance FPGA-based post quantum cryptography implementations, in: 2022 32nd International Conference on Field-Programmable Logic and Applications (FPL), 2022, pp. 456–457.
[33] S.J. B, V. Sankar, R.S. Lopez, V. K S, C.M. Stuart, A review on FPGA implementation of lightweight cryptography for wireless sensor network, in: 2023 International Conference on Power, Instrumentation, Control and Computing (PICC), 2023, pp. 1–6.
[34] I. El Gaabouri, M. Senhadji, M. Belkasmi, A survey on lightweight cryptography approach for IoT devices security, in: 2022 5th International Conference on Networking, Information Systems and Security: Envisage Intelligent Systems in 5G//6G-based Interconnected Digital Worlds (NISS), 2022, pp. 1–8.
[35] H.S. Lamkuche, D. Pramod, CSL: FPGA implementation of lightweight block cipher for power-constrained devices, Int. J. Inf. Comput. Secur. 12 (2020) 349–377.
[36] J.G. Pandey, A. Laddha, S.D. Samaddar, A lightweight VLSI architecture for RECTANGLE cipher and its implementation on an FPGA, in: 2020 24th International Symposium on VLSI Design and Test (VDAT), 2020, pp. 1–6.
[37] K. Kumar.V.G, A. Poojary, C.S. Rai, H.R. Nagesh, Implementation of lightweight cryptographic algorithms in FPGA, in: 2017 International Conference on Circuits, Controls, and Communications (CCUBE), 2017, pp. 232–235.
[38] A.A. Yazdeen, S.R.M. Zeebaree, M. Sadeeq, S.F. Kak, O.M. Ahmed, R.R. Zebari, FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review, Qubahan Academic Journal 1 (2021) 8–16.
[39] J. Buchmann, J. Ding, Post-Quantum Cryptography: Second International Workshop, PQCrypto 2008 Cincinnati, OH, USA October 17–19, 2008 Proceedings, 5299, Springer Science & Business Media, 2008.
[40] J.A. Buchmann, D. Butin, F. Göpfert, A. Petzoldt, Post-quantum cryptography: state of the art, in: The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday, 2016, pp. 88–108.
[41] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, R. Cammarota, Post-quantum lattice-based cryptography implementations: a survey, ACM Comput. Surv. (CSUR) 51 (2019) 1–41.
[42] Y. Todo, M. Morii, Bit-based division property and application to Simon family, in: Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20–23, 2016, Revised Selected Papers 23, 2016, pp. 357–377.
[43] E.R. Naru, H. Saini, M. Sharma, A recent review on lightweight cryptography in IoT, in: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2017, pp. 887–890.
[44] N. Zidarič, K. Mandal, G. Gong, M. Aagaard, The Welch-Gong stream cipher - evolutionary path, Cryptogr. Commun. (2023) 1–37, https://doi.org/10.1007/s12095-023-00656-0.
[45] X.-j. Zhao, T. Wang, An improved differential fault attack on Camellia, IACR Cryptol. ePrint Arch. 2009 (2009) 585.
[46] M.M. Kermani, R. Azarderakhsh, J. Xie, Error detection reliable architectures of Camellia block cipher applicable to different variants of its substitution boxes, in: 2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST), 2016, pp. 1–6.
[47] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, et al., Midori: a block cipher for low energy, in: Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21, 2015, pp. 411–436.
[48] X. Dong, Y. Shen, Cryptanalysis of reduced-round midori64 block cipher, Cryptol. ePrint Arch. (2016).
[49] J. Smith, A. Johnson, Block cipher QARMA with error detection mechanisms, in: Proceedings of the IEEE International Conference on Cryptography, London, UK, 2023, pp. 29–30.
[50] Zulkifli Yusof, Zuwairie Ibrahim, Ismail Ibrahim, K.Z.M. Azmi, Nor Aziz, Abdul Aziz, Nor Hidayati, Mohd. Mohamad, Distance evaluated simulated kalman filter for combinatorial optimization problems, ARPN J. Eng. Appl. Sci. 11 (2016) 4911–4916.
[51] S.A. Rahmad, Z. Ibrahim, Z.Md Yusof, Simulated Kalman filter with modified measurement, substitution mutation and hamming distance calculation for solving traveling salesman problem, in: Enabling Industry 4.0 through Advances in Mechatronics, Singapore, 2022, pp. 309–320.

Associate Professor Ir Dr Nurul Hazlina Noordin holds a Doctor of Philosophy (Ph.D.) in Electrical and Electronics Engineering from the University of Edinburgh, as evidenced by her academic profile. Currently, she is affiliated with Universiti Malaysia Pahang (UMP) Al-Sultan Abdullah, where she serves as an Associate Professor in the Faculty of Electrical and Electronics Engineering Technology. Throughout her career, Nurul Hazlina Noordin has made substantial contributions to her field, including expertise in Antenna Design, FPGA Implementation and Design, Adaptive Arrays, Beamforming Algorithms, and Engineering Education. Her work, under UMP STEM Lab, has received recognition and citations, reflecting the significance of her contributions to the academic community. Moreover, her dedication extends to both research and education, exemplified by her roles in academia and her commitment to nurturing future engineering professionals.

Phuah Soon Eu receives his B.Eng. in Electrical and Electronics Engineering from the Universti Malaysia Pahang Al-Sultan Abdullah in 2021. He is currently a product engineer at Intel (M) Sdn Bhd., Penang where he supported the high-volume manufacturing of the 13th and 14th generation of client CPUs. Phuah Soon Eu's research interest includes embedded system and electronic systems design. His work reflects a strong commitment to advancing the field of electrical engineering and electronics, as well as contributing to the ever-evolving landscape of technology. Phuah Soon Eu actively engages in mentorship roles, particularly as a mentor for the UMP STEM Lab outreach programs.

Associate Professor Dr Zuwairie Ibrahim received his B.Eng. in electrical engineering and an M.Eng. degree in image processing from Universiti Teknologi Malaysia, Johor, Malaysia, in 2000 and 2003, respectively. In 2006, he received his Ph.D. degree in DNA computing from Meiji University, Tokyo, Japan. From 2002 to 2008, he was a lecturer at Universiti Teknologi Malaysia, Johor, Malaysia. He was then promoted to senior lecturer in 2008 and served Universiti Teknologi Malaysia until 2012. In 2012, he transferred to Universiti Malaysia Pahang, Pahang, Malaysia, to work as an associate professor. He is one of the inventors of the SKF algorithm and SAFIRO. His research interests include the fundamentals and applications of computational intelligence, specifically, particle swarm optimization, ant colony optimization, gravitational search algorithm, and black hole algorithms.