IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 19, 2020, accepted May 5, 2020, date of publication May 11, 2020, date of current version May 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993600

A Hardware Descriptive Approach to Beetle Antennae Search

ZONGCHENG YUE[1], GANG LI[2], XIANGYUAN JIANG[3], SHUAI LI[4], (Senior Member, IEEE), JIAN CHENG[2], AND PENG REN[1], (Senior Member, IEEE)

[1]College of Oceanography and Space Informatics, China University of Petroleum (East China), Qingdao 266580, China
[2]Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[3]Institute of Marine Science and Technology, Shandong University, Qingdao 266237, China
[4]College of Engineering, Swansea University, Swansea SA1 8EN, U.K.

Corresponding author: Peng Ren (pengren@upc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Project U1906217, and in part by the Innovative Research Team Program for Young Scholars at Universities in Shandong Province under Project 2020KJN010.

ABSTRACT Beetle antennae search (BAS) is a newly developed meta-heuristic algorithm which is effectively used for optimizing objective functions of complex forms or even unknown forms. The common practice for implementing meta-heuristic algorithms including the BAS largely relies on programming in a high-level language and executing the code on a computer platform. However, the high-level implementation of the BAS algorithm hinders it from being used in an embedding system, where real-time operations are normally required. To address this limitation, we present an approach to implementing the BAS algorithm on a field-programmable gate array (FPGA). Specifically, we program the BAS function in the Verilog hardware description language (HDL), which provides a tractable vehicle for implementing the BAS algorithm at the gate level on the FPGA chip. We simulate our Verilog HDL based BAS module with the Modelsim platform. Simulation results validate the feasibility of our proposed Verilog HDL implementation of the BAS. Additionally, we implement the BAS model on the Zynq XC7Z010 platform, with 132.5 μ s latency for model implementation.

INDEX TERMS FPGA, beetle antennae search, Verilog HDL, optimization algorithm.

I. INTRODUCTION

A large number of meta-heuristic algorithms are designed by simulating certain biological behaviors. Though most of the biological behavioral functionalities are heuristic, they exhibit great explorative power that derives insights into effective paradigms for solving complex optimization problems. One biological organism may not clearly know all the situations in a strange environment. However, the organism is capable of using its explorative characteristics to find the conditions that are most conducive to its own survival. In light of this observation, meta-heuristic algorithms are developed subject to the explorative intrinsics of organisms. Such explorative properties enable the meta-heuristic algorithms to be effective in optimizing objective functions of complex or even unknown forms. In the research literatures, a considerable number of meta-heuristic algorithms including the gray wolf optimizier (GWO) [1], the genetic algorithm (GA) [2] and the particle swarm optimization (PSO) [3] are widely used in various practical optimization problems. Additionally, the beetle antennae search (BAS) is a newly developed meta-heuristic algorithm, which solves optimization problems with simple structure and results in solutions comparable to or superior to existing algorithms. The BAS algorithm has been widely used in a large number of practical projects. Wu *et al.* [8] used BAS algorithm to optimize obstacle avoidance systems of UAV and robot separately. Xie *et al.* [11] [12], [13] applied BAS to the design and optimization of ship collision avoidance system and precise control of marine diesel engine. In addition, BAS is applied to the design of power system and the construction of stock investment model [9], [14], [16], [17]. Mu *et al.* [18] used the BAS algorithm in 3D path planning. Fan *et al.* [21] applied the algorithm to an electro-hydraulic position servo control system. Moreover, BAS plays an important role in algorithm optimization. Lei *et al.* [10] designed an improved pollen algorithm with better performance in some areas based on BAS. Qi *et al.* [22]

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Hao Chen.

used BAS for parameter optimization of Qubit neural tree networks. Lin *et al.* [15] further optimized the problems of BAS in high-dimensional problems. Wang *et al.* [19] made the BAS algorithm play an important role in the optimization of radar waveforms. Different from the methodological investigations regarding BAS, in this paper, we make an extensive implementational study of the BAS algorithm.

The implementations of meta-heuristic algorithms (e.g., the BAS) largely rely on programming in a high-level language and executing the code on a computer platform. However, the high-level implementation of the BAS algorithm hinders it from being used in an embedding system, where real-time operations are normally required. To address this limitation, we investigate how to implement the BAS at a gate circuit level. Specifically, we describe how to use the Verilog hardware description language (HDL) to program the BAS on a field-programmable gate array (FPGA) chip.

An FPGA is a reconfigurable and reprogrammable chip that is able to implement algorithms and processing procedures at a gate circuit level. An FPGA is of small size, but with high computation speed. For these reasons, FPGA based embedded systems are widely used in not only huge manufacturing equipments but also mobile devices. FPGA has obvious advantages in several aspects compared with the traditional hardware structure. The first is high speed with parallelism [34]. FPGA allows multiple module circuits to operate in parallel, greatly increasing operation speed. The second is dynamic partial reconfiguration [35], [36]. FPGA allows developers to reconfigure bit-level hardware architecture. It is a competitive feature especially for tasks that require high-precision control and require less storage resources. The third is low latency. The latency of FPGAs is generally at the magnitude of nanoseconds, in contrast to microseconds required for GPUs.

In order to explore the possibility of operating the BAS at a circuit level rather than programming it in a high-level language, we investigate the scheme of implementing the BAS on an FPGA. In the literature, how to implement traditional meta-heuristic algorithms on FPGAs has been investigated. To the best of our knowledge, we are the first to implement the BAS model on an FPGA platform. In comparison, genetic algorithms (GAs) are also widely involved for FPGA implementations to solve optimization problems [26], [32]. In terms of optimization procedures, GAs mainly use chromosomal inheritance to select appropriate offspring, and perform mutation and elimination in the offspring to compute optimal values. The BAS algorithm uses the antennae to sense odor strength for optimization. In terms of programming implementations, GAs directly use binary coding, which is easy to implement on FPGA platforms. In contrast, the FPGA-based BAS model we implement is optimized in a continuous space, which is more sophisticated to explore. Instead of using the binary encoding scheme that is adopted by GAs but would induce accuracy decease if used in BAS, we use eight-bit fixed-point numbers for encoding. Therefore, in our work, we present an FPGA implementational approach to the BAS which is different from that of GA. Specifically, the main contributions of our work are summarized as follows:

1) We are the first first to present an approach that implements the BAS algorithm on FPGA.
2) We build a twin model on the FPGA chip by assigning different seeds to generate two pseudo-random numbers for the algorithm.
3) We implement the program by using a fixed-point arithmetic scheme, which ensures the accuracy of the algorithm and saves computing resources.

To the best of our knowledge, we are the first to implement the BAS model on an FPGA platform. Specifically, we develop a twin model of LFSR along with a fixed-point arithmetic scheme, resulting a unique implementation approach to the BAS on the FPGA platform Zynq XC7Z010. Furthermore, we apply the unique implementation approach to optimizing the booth function [33], which is a widely used benchmark for testing evolutionary algorithms. Our BAS implementation approach has 132.5 μ s latency, which turns out to be much more efficient than executions of high-level language BAS code on a computer platform.

The structure of this paper is arranged as follows: Section II introduces the biological principle of the BAS and describes how the algorithm simulates the biological trajectory of the beetle. Section III presents execution flow of the BAS algorithm by pseudo codo. Section IV describes the principles and processes of fixed-point number [27] operations. Section V introduces the implementation of BAS by using Verilog HDL, and presents the main Verilog HDL source code for implementation. Section VI presents the RTL [5] circuit diagram of each module and analyzes data processing procedure in the module, and the correspondence between the hardware structure and the algorithm. Section VII gives experimental results and experimental evaluation. Section VIII concludes the paper.

II. THE BEETLE ANTENNAE SEARCH (BAS) ALGORITHM

The beetle antennae search (BAS) algorithm [4] is developed in terms of simulating a beetle's behavioral trajectory for foraging food. In a strange environment, the beetle uses the two antennae on its head to conduct a series of flying and landing behaviors for the foraging. In the initial stage of the foraging, the beetle does not know where the food source is, and its two antennae are oriented randomly. It guesses the direction of the food through the odorants received by the antennae. The beetle judges which antenna receives the stronger odor, and accordingly uses its orientation as the estimated direction of the food source. Then the beetle flies along the estimated direction for a certain distance and lands with the two antennae oriented randomly [24]. One cycle of the beetle antennae search consists of a directional flying procedure and a randomly directional landing procedure. The cycle is repeated until the final food source is located. Such repetitive procedures are illustrated in Figure 1. In Figure 1,

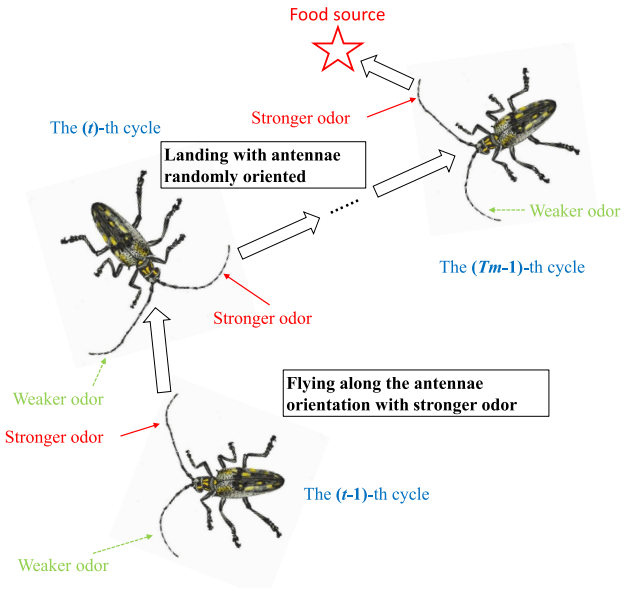


FIGURE 1. Repetitive procedures of the beetle antennae search for the food source.

one round for foraging until the food is found consists of a finite number of cycles. t-1 and t represent two sequential cycles in one round of foraging. T_m represents the the maximum number of cycles and it practically prevents the programs from endless execution.

In the optimization framework simulating the beetle antennae search, an objective function describes the relationship between the odor of food and the position of an beetle. We denote objective function by $f[\boldsymbol{x}(t)]$, where $\boldsymbol{x}(t)$ denotes the position of an beetle at the t-th time cycle. Specifically, the value of $f[\boldsymbol{x}(t)]$ implies the strength of the odor received by a beetle antenna located at $\boldsymbol{x}(t)$. The BAS aims at finding the optimal $\hat{\boldsymbol{x}}$ that minimizes the objective function f. The BAS algorithm is formulated in terms of repeating the cycle of directional flying and randomly directional landing until the optimal value of $f(\hat{\boldsymbol{x}})$ is obtained.

In order to make a consistent formulation, the randomly directional landing procedure is described ahead of the directional flying procedure. The detailed BAS formulation is presented as follows.

A. RANDOM ANTENNA ORIENTATIONS

The BAS algorithm commences by randomly generating a normalized directional vector that characterizes the orientations of the beetle antennae. Let $\boldsymbol{d}(t)$ be a normalized directional vector at t-th cycle, and it is generated as follows:

$$\boldsymbol{d}(t) = \frac{\boldsymbol{r}(t)}{\|\boldsymbol{r}(t)\|}, \tag{1}$$

where $\boldsymbol{r}(t)$ is a random vector which can be generated by programming in a high-level programming language.

The equation (1) is used for characterizing the orientation of an antenna randomly placed by the beetle.

B. ANTENNA POSITIONS WITH RESPECT TO RANDOMLY ORIENTED LANDING

Let $\boldsymbol{x}(t)$ denote the position of beetle when it lands at the t-th time cycle. Let $\boldsymbol{x}_r(t)$ and $\boldsymbol{x}_l(t)$ denote the position of the right and left antennae of a beetle at the t-th time cycle, respectively. Let $\rho(t)$ denote the sensing distance at the t-th time cycle, which keeps deceasing in the flying and landing cycles. The positions of the right and left antenna are computed as follows:

$$\begin{aligned} \boldsymbol{x}_r(t) &= \boldsymbol{x}(t) + \rho(t)\boldsymbol{d}(t), \\ \boldsymbol{x}_l(t) &= \boldsymbol{x}(t) - \rho(t)\boldsymbol{d}(t). \end{aligned} \tag{2}$$

Initially, the value of $\rho(t)$ is set to be large enough for the purpose of avoiding the local minima.

Based on (1), the equation (2) characterizes that the two antennae of the beetle are oriented randomly. The right antenna follows the positive direction along $\boldsymbol{d}(t)$ and the left follows the negative direction along $\boldsymbol{d}(t)$.

C. BEETLE POSITION SUBJECT TO DIRECTIONAL FLYING

According to the biological nature of a beetle, it chooses the antenna orientation with stronger odor as the flying direction. In the light of this observation, the flying direction is computed as follows:

$$\boldsymbol{d}_f(t) = \boldsymbol{d}(t)\text{sign}(f[\boldsymbol{x}_r(t)] - f[\boldsymbol{x}_l(t)]). \tag{3}$$

In (3), the bigger value between $f[\boldsymbol{x}_r(t)]$ and $f[\boldsymbol{x}_l(t)]$ is used to determine flying direction.

The beetle then flies long the direction $\boldsymbol{d}_f(t)$ and lands at a new position:

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \boldsymbol{d}_f(t)\xi(t), \tag{4}$$

where $\xi(t)$ denotes the flight distance of the beetle at t-th cycle. Initially, the value of $\xi(t)$ is set to be large enough for the purpose of avoiding local minima.

D. BEETLE POSITION DETERMINATION

The BAS algorithm makes a comparison between the state $\{\boldsymbol{x}(t+1), f[\boldsymbol{x}(t+1)]\}$ at the new position and the current state $\{\hat{\boldsymbol{x}}, f(\hat{\boldsymbol{x}})\}$. The process of the comparison is given as follows:

$$\{f(\hat{\boldsymbol{x}}), \hat{\boldsymbol{x}}\} = \begin{cases} \{f[\boldsymbol{x}(t+1)], \boldsymbol{x}(t+1)\} & f[\boldsymbol{x}(t+1)] < f(\hat{\boldsymbol{x}}); \\ \{f(\hat{\boldsymbol{x}}), \hat{\boldsymbol{x}}\} & f[\boldsymbol{x}(t+1)] > f(\hat{\boldsymbol{x}}). \end{cases} \tag{5}$$

Each obtained $f[\boldsymbol{x}(t+1)]$ is compared with the current $f(\hat{\boldsymbol{x}})$ after each cycle. If the $f[\boldsymbol{x}(t+1)]$ is smaller, $f[\boldsymbol{x}(t+1)]$ and $\boldsymbol{x}(t+1)$ replace $f(\hat{\boldsymbol{x}})$ and $\hat{\boldsymbol{x}}$, respectively, otherwise $f(\hat{\boldsymbol{x}})$ and $\hat{\boldsymbol{x}}$ keep the original values and $f[\boldsymbol{x}(t+1)]$ and $\boldsymbol{x}(t+1)$ are ignored.

E. CONVERGENCE FACTOR UPDATES

The convergence factors are updated as follows:

$$\rho(t+1) = 0.95\rho(t) + 0.01, \tag{6}$$

$$\xi(t+1) = 0.95\xi(t). \tag{7}$$

The equation (6) simulates that the sensing distance of the beetle decreases as the beetle approaches the food source. The equation (7) simulates that the flying distance of the beetle decreases as the beetle approaches the food source.

III. ALGORITHMIC DESCRIPTION OF BAS

Algorithm 1 describes how the beetle antennae search (BAS) executes in terms of pseudo code.

Algorithm 1 Beetle Antennae Search

Input: The objective function $f(\cdot)$, the initial convergence factors $\xi(0)$ and $\rho(0)$, the random vector $\boldsymbol{r}(t)$, the maximum number of cycles T_m

Output: The optimal $\hat{\boldsymbol{x}}$ that minimizes $f(\cdot)$

while $t < T_m$ **do**
- Generate the normalized directional vector $\boldsymbol{d}(t)$ according to (1);
- Compute the locations of the two randomly oriented antennae according to (2);
- Compute the flying direction according to (3);
- Update the time step index $t = t + 1$;
- Compute the landing location of the beetle according to (4);
- **if** $f[\boldsymbol{x}(t+1)] < f(\hat{\boldsymbol{x}})$ **then**
  - $f[\hat{\boldsymbol{x}}] = f[\boldsymbol{x}(t+1)], \hat{\boldsymbol{x}} = \boldsymbol{x}(t+1)$;
- Update sensing distance $\rho(t+1)$ and flight distance $\xi(t+1)$ according to (6) and (7), respectively;

return $\hat{\boldsymbol{x}}$ and $f(\hat{\boldsymbol{x}})$.

IV. FIXED-POINT ARITHMETIC

The BAS is normally implemented in high-level programming languages such as Matlab. High-level programming languages tend to manipulate decimals easily. However, high-level implementation of BAS requires computational overheads and cannot guarantee real-time operations. When BAS is implemented in an embedded system, it is very likely that the computational resources are limited but the computational efficacy is highly demanded. In this scenario, how to implement BAS in a low (hardware) level language is worth investigating. To this end, we investigate how to implement BAS in Verilog hardware description language (HDL). The Verilog HDL code renders register transfer level(RTL) implementation of BAS based on a field-programmable gate array(FPGA) chip. Specially, Verilog HDL builds gate-level circuits, which are all binarily coded. On the other hand, the implementation of the BAS algorithm requires operating continuous variables in a continuous space. There are two ways to characterize continuous decimals in FPGA operational processes, i.e., fixed-point operations and floating-point operations [23], [29].

Floating-point operations are more accurate than fixed-point operations [28]. However, the floating-point operations consume huge FPGA hardware resources [6]. On the other hand, the fixed-point operations consume fewer hardware resources with a guarantee of acceptably accurate results. According to the advantages of fixed-point operations, we use fixed-point arithmetic in our work. We give an example of 5.555 times 4.444 to explain the algorithmic principle of fixed-point operations.

Firstly, two numbers 5.555 and 4.444 are multiplied by two to the eight:

$$5.555 * 2^8 = 1422.08 \tag{8}$$

$$4.444 * 2^8 = 1137.664\,. \tag{9}$$

The two computed values are rounded, resulting in the decimal values 1422 and 1137, separately. We write them into the Verilog HDL decimal forms, i.e., 20′d1422 and 20′d1137, separately. Accordingly, we also write them into the Verilog HDL hexadecimal forms, i.e., 20′h5_8E and 20′h4_71, separately. The form of a Verilog HDL number consists of three parts, which are data width, decimal indication and value of the data. Take the number 20′d1422 as an example. '20' denotes the data width in a binary form, and it allows the maximum value of the data to be 2^{20}. ''d' indicates that the number belongs to the decimal system. '1422' denotes the value of the data, and it is represented in the decimal form. Similarly, in 20′h4_71, ''h' indicates that the number belongs to the hexadecimal system. '_' is a virtual symbol that denotes the fixed decimal point. It does not change the value of the data but virtually marks the location of the decimal point in our work.

The decimal point position of the fixed point number is fixed. One key technology of fixed-point arithmetic is to determine the fixed position, which is set by the designer of the decimal point. We obtain an intermediate value by multiplying a number by a specific integer power of two. For binary numbers, this operation is equivalent to shifting the **decimal point** to right by a specific integer. We then use the intermediate value to do other operations to obtain an intermediate result. After that, the intermediate result is divided by an integer power of two to obtain the final result. For binary numbers, this is equivalent to shifting the **entire number bits** to right by a specific integer. During data bits shift to right, extra data bits are rounded with respect to the fixed data width. Large data width renders a big shift of the decimal point and accordingly accurate data representation. However, too many data bits lead to a large amount of computational consumption. In the light of the observation, we use eight fractional part bits in our work to compromise the accuracy and the bit width of the data.

After (8) and (9), we shift the entire number bits to right by eight. Each hexadecimal number is represented by a four-bit binary number. We shift the decimal point to left by two bits in the hexadecimal number to keep the value of the data constant by using the symbol '_'. This is equivalent to that the decimal point is shifted to left by eight bits in the binary number. We thus put the symbol '_' there. Compared with

the decimal before rounding, the errors of the two numbers are 0.0056% and 0.058%, separately.

Secondly, the two intermediate quantities are multiplied as follows:

$$20'h5\_8E * 20'h4\_71 = 20'h18\_ABAE; \tag{10}$$

where $*$ denotes the operation of multiplication. The result in (10) is an intermediate one in the overall computation.

Thirdly, we divide the intermediate result by 256 (i.e., 2^8). In the binary operation, the original number is shifted to right by eight bits.

$$20'h18\_ABAE \ggg 8 = 20'h18\_AB; \tag{11}$$

The symbol '$\ggg$' denotes the right shift operation.

Fourthly, the obtained value is converted into a decimal system number as follows:

$$20'h18\_AB = 24.171. \tag{12}$$

Fifthly, in order to make a comparison between the Verilog HDL computation and the actual result, we compute the actual value as follows:

$$5.555 * 4.444 = 24.68642. \tag{13}$$

According to the results of (12) and (13), we compute the error in the fixed-point arithmetic. The computational formula is as follows:

$$\frac{24.68642 - 24.171}{24.68642} \approx 0.0208. \tag{14}$$

The error is about 2%. The error is caused by the operations that the fixed-point number has an approximation and rounding. In fact, the operations cause a small error for a number with long width. We reduce the error by setting a higher bit width.

V. VERILOG HDL IMPLEMENTATION OF BAS

In this section, we introduce the approach to implementing BAS by Verilog HDL. Specifically, Sections V-A, V-B, V-C and V-E correspond to the Sections II-A, II-B, II-C and II-E, respectively.

A. RANDOM DIRECTION GENERATION BY LINEAR FEEDBACK SHIFT REGISTER (LFSR)

The randomization process includes the randomization of the evolution direction and the randomness of the motion direction. It is a very important process for the BAS algorithm.

When the algorithm is implemented in high-level programming languages, a random function embedded in the programming language is called to generate a random number. Although the Verilog HDL has a function that generates random numbers, the function is just available for external stimulus but not for circuit implementation. To implement the entire algorithm of BAS on a single chip, it is necessary to generate random numbers on the chip system. However, there is no reported manner to generate a completely random number inside the FPGA. We use pseudo-random numbers instead

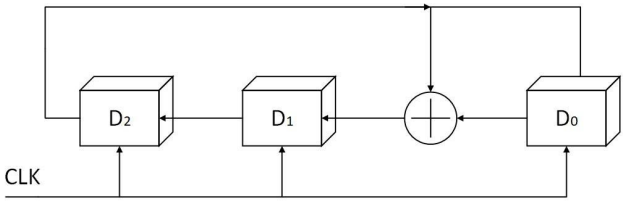


FIGURE 2. Structural model diagram of LFSR with three D flip-flops.

of random numbers. In order to make a clear presentation for implementational facilitation, we use the **standardized form** to represent Verilog HDL code.

We commence by giving the definition of the pseudo-random number. The pseudo-random number is a sequence of numbers generated by a given specific set of seeds according to a specific algorithm or structure. In fact, the number is generated by a specific algorithm. The result is completely known. However, there are large numbers of pseudo-random numbers, and the degree of randomness is completely sufficient. In our work, we mainly use the linear feedback shift register (LFSR) [25] to generate pseudo-random numbers.

The LFSR consists of D flip-flops and XOR gates. LFSR generates $2^N - 1$ 'random numbers'. As long as there are more D flip-flops, more 'random numbers' are generated and the numbers generated are more 'random'.

Figure 2 illustrates a simplified structural model of LFSR, where the D_2, D_1 and D_0 denote different digits of the same binary number. The CLK is a clock signal which activates the LFSR model. The clock signal is generated by crystal oscillations. The crystal is embedded in the FPGA. The symbol '$\oplus$' represents XOR operation, and its logical expression is $A \oplus B = \overline{A}B + A\overline{B}$ where $\overline{A}$ represents for the bitwise inversion of the binary form of A.

We use an example to explain the operating mechanism of LFSR. Each binary number generated by LFSR is a state. The initial state is set by the given seed according to the definition of pseudo-number. We set the initial seed to be 111. The initial state is $D_2\ D_1\ D_0 = 111$(seed) where the seed is an initial state of the entire LFSR system. The subsequent state is

$$\begin{aligned} D_2 &= D_1 = 1, \\ D_1 &= D_0 \oplus D_2 = 0, \\ D_0 &= D_2 = 1, \end{aligned} \tag{15}$$

where $D_2\ D_1\ D_0 = 101$ according to Figure 2.

LFSR repeats the above process. We obtain a state transition diagram as illustrated in Figure 3. Figure 3 is a schematic diagram of the LFSR state transition, and it is generated by the LFSR model. The LFSR model stores each bit of a N bit number in a flip-flop. The input of each flip-flop is the output of the previous flip-flop or the XOR of the other two flip-flops' outputs. When one-bit changes, the whole number changes.

As shown in Figure 2 and Figure 3, we use three D flip-flops. There are seven states in total for the three D flip-flops.

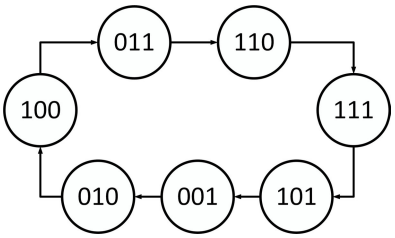


FIGURE 3. The states transition diagram.

In our work, we suppose that a beetle should move in a two-dimensional environment. The beetle requires two random numbers for providing two random directions for the two-dimensional environment. However, the LFSR module generates only one pseudo-random number. To address the deficiency, we build a twin model to generate two pseudo-random numbers. The twin model needs two seeds. In order to avoid the two pseudo-random numbers generated exactly the same, we give two different seeds to provide multiple data combinations for the generated pseudo-random numbers.

The initial state of the twin model is given by two totally different seeds, and the activation signal is given by the CLK signal.

We use horizontal and vertical positions to represent the two-dimensional environment. Let dir_x1 represent the horizontal direction and dir_x2 represent the vertical direction in the Verilog HDL statement. Based on the LFSR model, we give two different seeds to generate dir_x1 and dir_x2.

B. UPDATING THE ANTENNA POSITIONS WITH FIXED POINT VERILOG HDL IMPLEMENTATION

In BAS algorithm, the convergence is related to distance decrease. Specifically, both the moving distance and the sensing distance are supposed to converge in the BAS algorithm [30]. The process of updating the factors is not a simple subtraction but a decrease of moving distance and sensing distance, which requires fractional operations.

Section II illustrates BAS algorithm in high-level programming language. According to (2), (4), (6) and (7), the sensing distance and the moving distance occur change from to cycle. In each cycle, the two parameters are updated. In the updated process, $\rho(t)$ which denotes the sensing distance in (6) and $\xi(t)$ which denotes the moving distance in (7) inevitably have a fractional part. However, Verilog HDL cannot perform fractional operations directly. To address this limitation, we use fixed point operation in Verilog HDL according to Section IV.

We develop a function f(x1, x2) based on Verilog HDL, where x1 and x2 are inputs of the function. The Verilog function f(x1, x2) characterizes an objective function $f(\boldsymbol{x})$ with a two-dimensional vector variable $\boldsymbol{x}$, and x1 and x2 represent the two elements of $\boldsymbol{x}$, separately. x1 and x2 are wire type data whose bit width is set to be 16 bits. The output is the function f(x1, x2) itself and it is reg type data whose bit is set to be 24 bits. The wire and the reg are two data types defined by Verilog HDL. The operation process of the BAS with the fixed-point arithmetic method is in the following subsection.

The position coordinates of the current antennae are computed according to the random direction. The position coordinates of the current antennae are described in Verilog HDL as follows:

$$\begin{aligned} \text{xl\_1} &= \text{x1} + (\text{dir\_x1} * \text{sense}) \ggg 8; \\ \text{xl\_2} &= \text{x2} + (\text{dir\_x2} * \text{sense}) \ggg 8; \end{aligned} \tag{16}$$

$$\begin{aligned} \text{xr\_1} &= \text{x1} - (\text{dir\_x1} * \text{sense}) \ggg 8; \\ \text{xr\_2} &= \text{x2} - (\text{dir\_x2} * \text{sense}) \ggg 8; \end{aligned} \tag{17}$$

where xl_1 denotes the horizontal position of the beetle left antenna, xl_2 denotes the vertical position of the beetle right antenna, and x1 and x2 represent the current horizontal and vertical coordinates of the beetle, respectively.

In Section II, the value of f (f is a scaler and it's a value that computed based on $\boldsymbol{x}(t)$) implies the odor received by the antenna located at $\boldsymbol{x}(t)$ which represents a vector in high-level program language. In Verilog HDL, We decompose vectors into scalars. For example, the value of f implies the odor received by the antenna in (18), and the position of the beetle becomes (x1, x2), where x1 and x2 are scalars. f_l denotes the odor that the beetle left antenna receives. f_r denotes the odor the beetle right antenna receives. f_l and f_r are computed in terms of Verilog HDL as follows:

$$\begin{aligned} \text{f\_l} &= \text{f(xl\_1, xl\_2)}; \\ \text{f\_r} &= \text{f(xr\_1, xr\_2)}; \end{aligned} \tag{18}$$

where calculation for f is based on the position coordinates of the current antennae. The beetle decides to go towards a direction which is characterized the larger one between f_l and f_r.

C. UPDATING THE BEETLE POSITION IN TERMS OF VERILOG HDL IMPLEMENTATION

Let move denote the moving distance. The Verilog HDL implementation of computing the current moving distance is given in terms of Verilog HDL as follows:

$$\begin{aligned} \text{move\_x1} &= (\text{move} * \text{dir\_x1}) \ggg 8; \\ \text{move\_x2} &= (\text{move} * \text{dir\_x2}) \ggg 8; \end{aligned} \tag{19}$$

where move_x1 denotes the beetle moving distance along the horizontal direction, and move_x2 denotes the beetle moving distance along the vertical direction.

According to (16), (17), (18) and (19), the position updates for the beetle are implemented in Verilog HDL as follows:

$$\begin{aligned} \text{x1} &= \text{x1} - \text{move\_x1} * \text{sign(f\_l, f\_r)}; \\ \text{x2} &= \text{x2} - \text{move\_x2} * \text{sign(f\_l, f\_r)}; \end{aligned} \tag{20}$$

where the sign denotes a symbolic function indicating that the subsequent movement direction follows the larger one of the two values f_l and f_r.

D. UPDATING THE BEST POSITION IN TERMS OF VERILOG HDL IMPLEMENTATION

The first f on the left-hand of the statement (21) denotes the value of f at (x1, x2), and the second f denotes the objective function.

$$\mathrm{f} = \mathrm{f(x1, x2);} \tag{21}$$

We obtain (x1, x2) based on the statement (20). We send (x1, x2) to the objective function to obtain the corresponding f. Then, we compare f and f_best which is the current optimal value of the objective function. The process of comparison is given in terms of Verilog HDL as follows:

$$\begin{aligned}&\mathrm{if(f\_best} < \mathrm{f)}\\&\mathrm{begin}\\&\mathrm{x1\_best} = \mathrm{x;}\\&\mathrm{x2\_best} = \mathrm{y;}\\&\mathrm{f\_best} = \mathrm{f;}\\&\mathrm{end}\end{aligned} \tag{22}$$

where x1_best and x2_best denote the current optimal x1 and x2, respectively.

When the value of f is larger than the value of f_best, x1_best and x2_best are overwritten by the values of x1 and x2, respectively, and f_best is overwritten by f. When the value of f is smaller than the value of f_best, x1_best, x2_best and f_best keep the original values without updates.

E. UPDATING THE CONVERGENCE FACTORS IN TERMS OF VERILOG HDL IMPLEMENTATION

The convergence factors enable the BAS implemented in Verilog HDL to avoid local minima. sense_c and move_c denote initial value of the sensing distance and moving distance, respectively. They are given in terms of Verilog HDL as follows:

$$\begin{aligned}\mathrm{sense\_c} &= 0.95 * 2\hat{}8;\\ \mathrm{move\_c} &= 0.95 * 2\hat{}8;\end{aligned} \tag{23}$$

Here the data is multiplied by two to the eight for the purpose of fixed-point arithmetic operation in subsequent step.

Let sense denote the sensing distance. We multiply the initial value sense_c by the current value sense to shrink the sensing distance. We multiply the initial value move_c by the current value move to shrink the moving distance.

$$\begin{aligned}\mathrm{sense} &= (\mathrm{sense\_c} * \mathrm{sense}) \ggg 8;\\ \mathrm{move} &= (\mathrm{move\_c} * \mathrm{move}) \ggg 8;\end{aligned} \tag{24}$$

where the convergence factors are shifted to right by eight bits to obtain the sensing distance and moving distance of the next cycle.

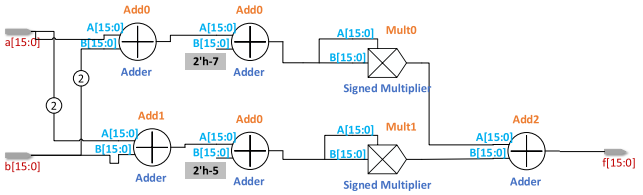


FIGURE 4. The RTL of the test function.

VI. CIRCUIT SYNTHESIS FOR BAS

In this section, the BAS algorithm implemented by Verilog HDL is divided into five modules. Firstly, the objective function module which is described in Section VI-A is generated. Secondly, the random number generator module and the twin module in Section VI-B are built. Finally, these modules are connected to each other such that the BAS system module is constructed in Section VI-E.

In each module, the circuit is generated, and the circuit is analyzed by describing how the modules work.

A. THE OBJECTIVE FUNCTION MODULE

The test function module circuit is generated as Figure 4.

The objective function is chosen as booth function [33], which is a widely used benchmark for testing evolutionary algorithms:

$$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2. \tag{25}$$

The circuit for the objective function is generated above all. The circuit which has two input lines is designed. a and b in Figure 4 are two input nodes of the input lines. The coordinates of the beetle (x1, x2) are sent to the two nodes. x1 sends to a and x2 sends to b.

The circuit computes the value of f based on (x1, x2). f implies the odor the beetle receives. The moving direction of the beetle is decided with respect to by f value.

B. THE RANDOM NUMBER GENERATOR MODULE

The random number generator module circuit is generated as follows:

According to the Section V-A, a twin LFSR model is built. The model is constructed by two single LFSR models of which consists of D flip-flops and XOR gates. The two models are given the same sensitive signal CLK according to Figure 2. We add the load signal load to the twin model to control the model startup. The reset signal rst_n is added to the twin model. The reset signal is responsible for setting the data to be zero to facilitate the debugging of the model. This module is designed to generate two pseudo-random numbers.

C. THE ANTENNAE MODULES

The left and right antenna module circuits are generated as follows:

The antenna modules consist of the left antenna register transfer level(RTL) model and the right antenna RTL model. They have similar struct according to Figure 6 and Figure 7.

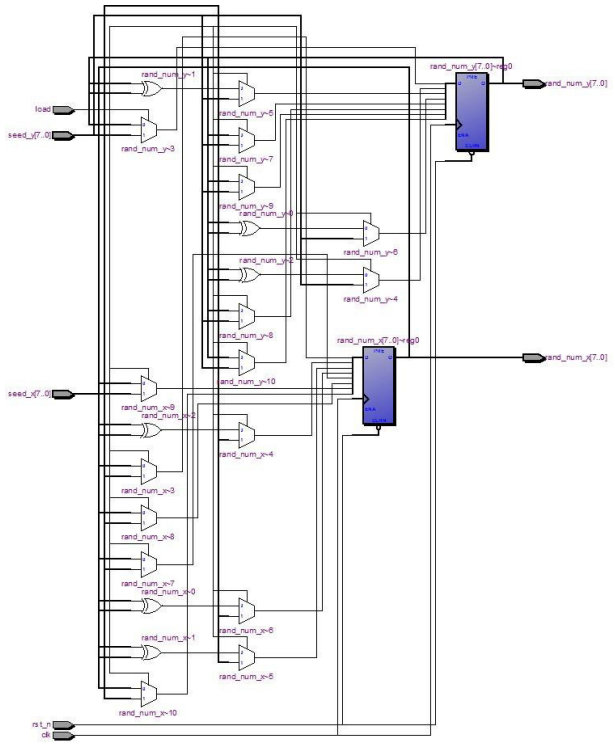


FIGURE 5. The RTL of the random number generator module.

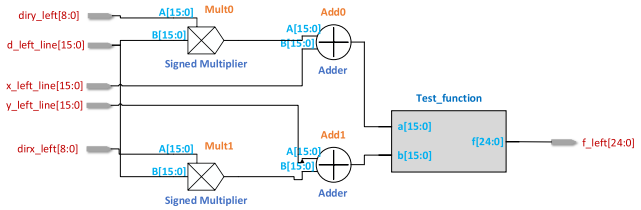


FIGURE 6. The RTL model of the beetle left antenna.

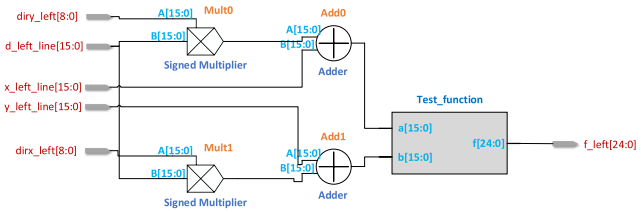


FIGURE 7. The RTL model of the beetle right antenna.

The only difference between them is the direction operation based on (16) and (17).

The left antenna RTL model and the right antenna RTL model obtain the coordinates (xl_1, xl_2) and (xr_1, xr_2), respectively. The coordinates are send to the objective function described in Section VI-A to compute the odor the two antennae received f_l and f_r. The modules are designed to obtain f_l and f_r.

D. THE BEETLE POSITION MODULE

The coordinate values in this module are updated, and f value is computed based on the coordinate values.

TABLE 1. Resource utilization of the BAS model on Zynq XC7Z010.

Resource	Utilization	Availability	Utilization%
LUT	877	17600	4.98
FF	16	35200	0.05
DSP	12	80	15
BFUG	1	32	3.13

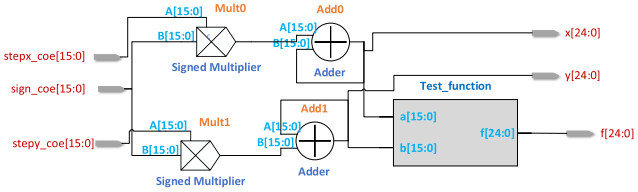


FIGURE 8. The RTL of the beetle position module.

The coordinate value of the previous position of the beetle is input, and the coordinate of the next position are computed as the output according to the coordinate value of the previous position.

The position coordinate of the beetle is updated. The position coordinate of the beetle is the input of the objective function to compute f. After f is obtained, the last f is compared. The larger f computed based on (x1, x2) becomes the new optimal result, otherwise, it is discarded. The above process is repeated until the results converge. The result obtained after convergence is reached as the final f. This module is designed to update the position of the beetle (x1, x2), and the final f.

E. THE OVERALL STRUCTURE OF THE BAS MODULE

Figure 9 illustrates the overall structure of the BAS module. The logic part which is composed of 4 modules performs computing tasks. In fact, the logic part is implemented as combinatorial logic. The twin LFSR module generates the random number for the antenna modules. The antenna modules uses random numbers to generate corresponding sense coordinates and sends them to the target function module. The target function module generates a function value, and passes this value to the beetle position module. The beetle position module updates the beetle coordinate according to the function value, and sends this coordinate to the target function module again. The target function module generates the final function value and sends it back to the beetle position module. The beetle position module sends the the finally generated function value to BRAM, and compares it with the current optimal value to decide whether to update the optimal value. The logic part repeats the process until reach the maximum iterations.

VII. EVALUATION AND EXPERIMENTS

We build the FPGA-based BAS model to find the optimal value of our objective function. The result shows that the model finds the optimal value by costing 132.5us. We give our resource consumption in Table 1.

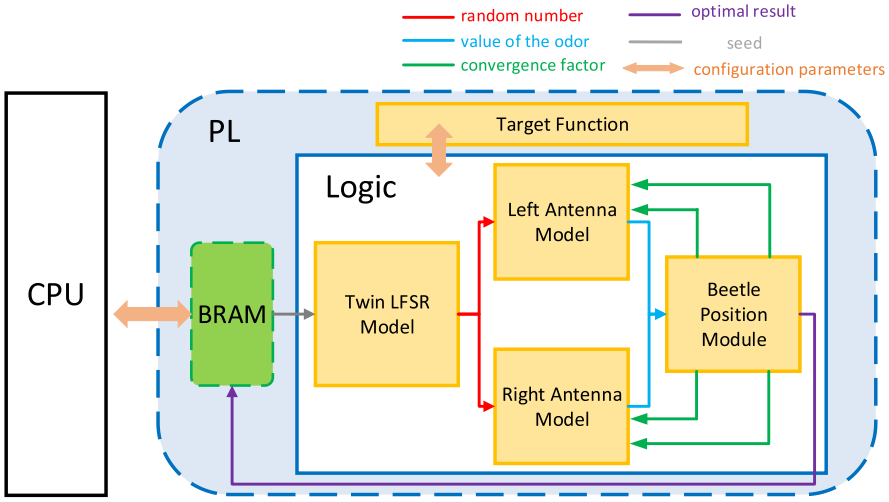


FIGURE 9. The beetle dataflow model.

A. RESULTS VERIFICATION

On the basis of the code being implemented, we use the ModelSim simulation to verify the correctness of the resulted values. We program the testbench to give the FPGA-based BAS algorithm an initial stimulus. The simulation results are given in Figure 10. Figure 10(a) illustrates the entire system data optimization process. The horizontal axis represents the number of iterations. The function value refers to the leftmost vertical axis, and the coordinate refers to the rightmost vertical axis. According to Figure 10, the coordinate value does not decrease all the time. The x1 and x2 increase and decrease alternately. The function value corresponding to the coordinate value continuously decreases, because this is the decisive condition for the system to find the optimal value.

We observe that the difference in the starting position affects the number of iterations in the optimization process during the experiment. To further verify this phenomenon, we randomly set several different starting positions for comparative experiments. The experimental results are given in Figures 10(b), 10(c) and 10(d).

Figure 10(b) illustrates that the starting position is set to (195, 177), and the number of iterations is reduced by about ten compared with Figure 10(a). Similarly, Figure 10(d) indicates that the starting position is set to (20, 3), and the number of iterations is reduced by about forty compared with Figure 10(a). We observe that a reasonable choice of starting position is beneficial to the optimization computation time. We select the first experimental result as the final experimental result for presentation. How to choose a reasonable starting position will continue to be studied in our future work.

B. HARDWARE IMPLEMENTATION

The overall hardware structure is implemented as described in Section VI. The CPU writes configuration parameters to the BRAM on the PL through the AXI GPIO. The configuration parameters include *rst_n* signal, *load* signal and *seed* stimulus. The CPU sends an interruption signal to the PL after the configuration parameters writing is completed. The PL refers the configuration parameters from the BRAM to the LFSR module. The LFSR module generates a set of random numbers for the BAS model. Each time when the BAS model produces a set of optimization results including coordinate values and their corresponding function values, it is rewritten to the BRAM. The CPU reads the optimized model from the BRAM when the maximum number of cycles is reached. The optimized data is finally read from BRAM, and the results are completely consistent with the simulation results.

We evaluate the BAS model by implementing it on Zynq XC7Z010 FPGAs, which contains a Cortex-A9 device, 866MHz dual-core ARM-based CPU and 1GB DDR memory. The overall system is developed using Verilog HDL and implemented with Vivado 2019.1, which performs synthesis and implementation. Our FPGA-based BAS model is executed at 100 MHz clock frequency. A reasonable clock frequency is conducive to the whole design. In fact, most of the Verilog code described in the Section V is implemented as combinatorial logic which causes huge power consumption at high frequencies. However, even with a huge amount of combinatorial logic, the FPGA-based BAS algorithm still consumes acceptable power. The BAS model is greatly concise. For example, at the working frequency of 100MHz, the power consumed is 1.64W, and at the working frequency of 150MHz, the power consumed is 1.729W. As our study is the first preliminary work for the BAS FPGA implementation, we use the default frequency, which affords acceptable efficiency.

C. RESOURCE UTILIZATION

Table. 1 presents the resource utilization of BAS model running on the Zynq XC7Z010 device at 100 MHz. A small amount of logical resources on the chip are required for

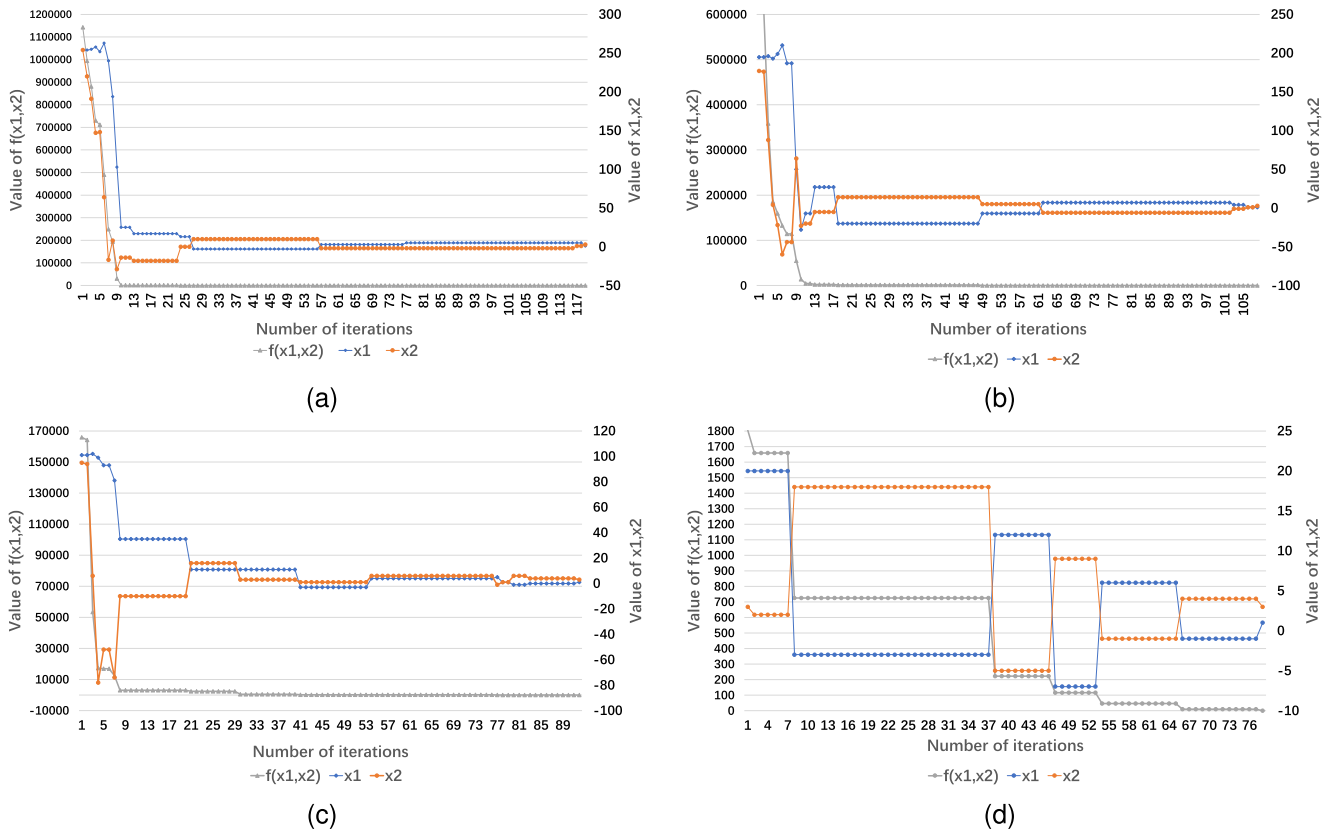


FIGURE 10. Data optimization for 4 different starting positions. (a) Starting position:(254, 254), iterations:121. (b) Starting position:(195, 177), iterations:109. (c) Starting position:(101, 95), iterations:92. (d) Starting position:(20, 3), iterations:78.

implementing the BAS model. This benefits from the simple structure of the BAS model.

D. DEGREE OF PARALLELISM

We duplicate multiple arithmetic circuits and compute them in parallel. The multiple arithmetic circuits are characterized (initialized) via different data seeds and this renders faster optimization convergence. The architecture of the BAS model is simple, and its operation time is short. The time benefitting from the system-level parallel computation of the BAS model is small but at the cost of increased resource consumption. Additionally, the twin LFSR module continuously generates random numbers. These numbers are not independent of one another and cannot be generated in parallel. Therefore, we do not conduct system-level parallel computation but parallelly update the convergence factor at the module level.

VIII. CONCLUSIONS

In this paper, a novel FPGA-based prototype has been developed for the BAS algorithm that mainly uses the two antennae of the beetle to simulate the foraging process of the beetle. Specifically, we have designed arithmetic circuits for every part of the beetle. We have built the twin LFSR model to enable the prototype work in two-dimensional environment on the chip. We have used the fixed-point arithmetic to overcome difficulties in data discontinuity on FPGA platform, and made a trade-off between computing resources and data accuracy.

We have also designed corresponding arithmetic circuits for the booth function. This prototype is implemented on the Zynq XC7Z010 platform to optimize the booth function with 132.5 μ s latency, working at 100MHZ.

Our future work is summarized from the follow aspects. In the literature, there are few hardware implementation algorithms for high-dimensional optimization problems. In order to further expand our approach, we plan to increase the dimension of the environment or function and explore the optimal performance of our method in a high-dimensional environment in future work. In addition, we found that choosing a reasonable initial position is beneficial to the function optimization time during the experiment. We will continue to pay attention to this part in future work.

REFERENCES

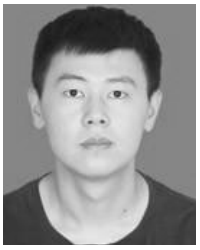
[1] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[2] P. Mazumder and E. M. Rudnick, *Genetic Algorithms: For Vlsi Design, Layout & Test Automation*. Reading, MA, USA: Addison-Wesley, 1999.

[3] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Piscataway, NJ, USA, Nov./Dec. 1995, pp. 1942–1948.

[4] X. Jiang and S. Li, "BAS: Beetle antennae search algorithm for optimization problems," 2017, *arXiv:1710.10724*. [Online]. Available: http://arxiv.org/abs/1710.10724

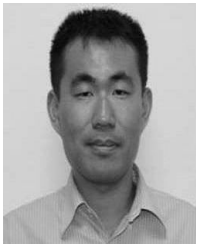
[5] T. Zhang, J. Wang, S. Guo, and Z. Chen, "A comprehensive FPGA reverse engineering tool-chain: From bitstream to RTL code," *IEEE Access*, vol. 7, pp. 38379–38389, 2019.
[6] C. Lammie, A. Olsen, T. Carrick, and M. Rahimi Azghadi, "Low-power and high-speed deep FPGA inference engines for weed classification at the edge," *IEEE Access*, vol. 7, pp. 51171–51184, 2019.
[7] S. Hou, Y. Guo, and S. Li, "A lightweight LFSR-based strong physical unclonable function design on FPGA," *IEEE Access*, vol. 7, pp. 64778–64787, 2019.
[8] Q. Wu, H. Lin, Y. Jin, Z. Chen, S. Li, and D. Chen, "A new fallback beetle antennae search algorithm for path planning of mobile robots with collision-free capability," *Soft Comput.*, vol. 24, no. 3, pp. 2369–2380, May 2019.
[9] Z. Cao, L. Liu, B. Hu, and H. Xie, "Short-term load forecasting based on variational modal decomposition and optimization model," in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 121–126.
[10] M. Lei, Q. Luo, Y. Zhou, C. Tang, and Y. Gao, "BFPA: Butterfly strategy flower pollination algorithm," in *Proc. Intell. Comput. Theories Appl. (ICIC)*, 2019, pp. 739–748.
[11] S. Xie, X. Chu, M. Zheng, and C. Liu, "Ship predictive collision avoidance method based on an improved beetle antennae search algorithm," *Ocean Eng.*, vol. 192, Nov. 2019, Art. no. 106542.
[12] S. Xie, V. Garofano, X. Chu, and R. R. Negenborn, "Model predictive ship collision avoidance based on Q-learning beetle swarm antenna search and neural networks," *Ocean Eng.*, vol. 193, Dec. 2019, Art. no. 106609.
[13] S. Xie, X. Chu, C. Liu, and M. Zheng, "Marine diesel engine speed control based on adaptive state-compensate extended state observer-backstepping method," *Proc. Inst. Mech. Eng., I, J. Syst. Control Eng.*, vol. 233, no. 5, pp. 457–471, May 2019.
[14] S.-W. Fei and C.-X. He, "Prediction of dissolved gases content in power transformer oil using BASA-based mixed kernel RVR model," *Int. J. Green Energy*, vol. 16, no. 8, pp. 652–656, Jun. 2019.
[15] M.-J. Lin and Q.-H. Li, "A hybrid optimization method of beetle antennae search algorithm and particle swarm optimization," in *Proc. Int. Conf. Elect., Control, Automat. Robot. (ECAR)*, 2018, pp. 396–401.
[16] Q. Li, Z. Wang, and A. Wei, "Research on optimal scheduling of wind-PV-hydro-storage power complementary system based on BAS algorithm," *IOP Conf. Series, Mater. Sci. Eng.*, vol. 490, Apr. 2019, Art. no. 072059.
[17] T. Chen, Y. Zhu, and J. Teng, "Beetle swarm optimisation for solving investment portfolio problems," *J. Eng.*, vol. 2018, no. 16, pp. 1600–1605, Nov. 2018.
[18] Y. Mu, B. Li, D. An, and Y. Wei, "Three-dimensional route planning based on the beetle swarm optimization algorithm," *IEEE Access*, vol. 7, pp. 117804–117813, 2019.
[19] Q. Wang, M. Li, L. Gao, K. Li, and H. Chen, "Nature-inspired waveform optimisation for range spread target detection in cognitive radar," *J. Eng.*, vol. 2019, no. 20, pp. 6767–6771, Oct. 2019.
[20] Q. Wu, X. Shen, Y. Jin, Z. Chen, S. Li, A. H. Khan, and D. Chen, "Intelligent beetle antennae search for UAV sensing and avoidance of obstacles," *Sensors*, vol. 19, no. 8, p. 1758, Apr. 2019, doi: 10.3390/s19081758.
[21] Y. Fan, J. Shao, and G. Sun, "Optimized PID controller based on beetle antennae search algorithm for electro-hydraulic position servo control system," *Sensors*, vol. 19, no. 12, p. 2727, Jun. 2019, doi: 10.3390/s19122727.
[22] F. Qi, Q. Wang, and L. Xu, "Disease classification model based on Qubit neural tree networks," in *Proc. 9th Int. Conf. Inf. Technol. Med. Edu. (ITME)*. Piscataway, NJ, USA: IEEE, Oct. 2018, pp. 154–158.
[23] J. Cheng and T. Liu, "A variational reconstructed discontinuous Galerkin method for the steady-state compressible flows on unstructured grids," *J. Comput. Phys.*, vol. 380, pp. 65–87, Mar. 2019.
[24] H. Zhou, W. L. Ouyang, and J. Cheng, "Deep continuous conditional random fields with asymmetric inter-object constraints for online multi-object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 4, pp. 1011–1022, Apr. 2019.
[25] J. Lu, D. Liu, H. Li, C. Zhang, and X. Zou, "A fully integrated HF RFID tag chip with LFSR-based light-weight tripling mutual authentication protocol," *IEEE Access*, vol. 7, pp. 73285–73294, 2019.
[26] M. A. Vega-Rodriguez, R. Gutierrez-Gil, J. M. Avila-Roman, J. M. Sanchez-Perez, and J. A. Gomez-Pulido, "Genetic algorithms using parallelism and FPGAs: The TSP as case study," in *Proc. Int. Conf. Parallel Process. Workshops (ICPPW)*, 2005, pp. 573–579.
[27] B. Chen, J. Wang, H. Zhao, N. Zheng, and J. C. Principe, "Convergence of a fixed-point algorithm under maximum correntropy criterion," *IEEE Signal Process. Lett.*, vol. 22, no. 10, pp. 1723–1727, Oct. 2015.
[28] T. Li and W. X. Zheng, "New stability criterion for fixed-point state-space digital filters with generalized overflow arithmetic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 7, pp. 443–447, Jul. 2012.
[29] L. Harnefors, "Implementation of resonant controllers and filters in fixed-point arithmetic," *IEEE Trans. Ind. Electron.*, vol. 56, no. 4, pp. 1273–1281, Apr. 2009.
[30] Q. Wu, Z. Ma, G. Xu, S. Li, and D. Chen, "A novel neural network classifier using beetle antennae search algorithm for pattern classification," *IEEE Access*, vol. 7, pp. 64686–64696, 2019.
[31] P. Cortés-Antonio, J. Rangel-GonzÃĺlez, and L. A. Villa-Vargas, "Design and implementation of differential evolution algorithm on FPGA for double-precision floating-point representation," *Acta Polytechnica Hungarica*, vol. 11, no. 4, pp. 139–153, 2014.
[32] V. A. Qureshi, S. Hanmandlu, and B. Papachary, "FPGA based design and implementation of genetic algorithm using Verilog," in *Proc. IJERA Nat. Conf. Develop., Adv. Trends Eng. Sci.*, 2015, pp. 42–46.
[33] M. Jamil and X. S. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.
[34] R. Wiśniewski, G. Bazydło, and P.Szcześniak, "SVM algorithm oriented for implementation in a low-cost Xilinx FPGA," *Integr., VLSI J.*, vol. 64, pp. 150–194, Jan. 2019.
[35] R. Wisniewski, "Dynamic partial reconfiguration of concurrent control systems specified by Petri nets and implemented in Xilinx FPGA devices," *IEEE Access*, vol. 6, pp. 32376–32391, 2018.
[36] R. Wiśniewski, *Prototyping of Concurrent Control Systems Implemented in FPGA Devices*. Zielona Gora, Poland: Springer, 2017.



ZONGCHENG YUE received the B.Eng. degree from Nangchang Hangkong University, Nanchang, China. He is currently pursuing the M.Eng. degree in electronics and communication engineering with the China University of Petroleum, Qingdao, China. His current research interests include machine learning and embedded systems.

GANG LI received the B.E. degree in information engineering from Northwestern Polytechnical University, Xi'an, China, in 2013. He is currently pursuing the Ph.D. degree with the National Laboratory of Pattern recognition, Chinese Academy of Sciences. His research interests include image understanding, deep learning, and embedded systems.

XIANGYUAN JIANG received the B.E. degree in biomedical engineering from Shandong University, China, in 2006, and the Ph.D. degree in control science and control engineering from Shandong University, in 2012. He was a Lecturer with the China University of Petroleum. He was a Research Fellow with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He is currently an Associate Professor with the Institute of Marine Science and Technology, Shandong University. His current research interests include the adaptive control of source seeking, distributed estimation, and control.



SHUAI LI (Senior Member, IEEE) received the B.E. degree in precision mechanical engineering from the Hefei University of Technology, Hefei, China, in 2005, the M.E. degree in automatic control engineering from the University of Science and Technology of China, Hefei, in 2008, and the Ph.D. degree in electrical and computer engineering from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2014. He is currently an Associate Professor (Reader) with Swansea University, Wales, U.K., leading the Robotic Laboratory, conducting research on robot manipulation and impedance control, multi-robot coordination, distributed control, intelligent optimization and control, and legged robots. Dr. Li is the Founding Editor-in-Chief of the *International Journal of Robotics and Control* and the General Co-Chair of 2018 International Conference on Advanced Robotics and Intelligent Control.

JIAN CHENG received the B.S. and M.S. degrees from Wuhan University, Wuhan, China, in 1998 and 2001, respectively, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2004. From 2004 to 2006, he was Postdoctoral Fellow with the Nokia Research Center, Beijing. He has been with the National Laboratory of Pattern Recognition, Beijing, since 2006. His current research interests include machine learning methods and their applications for image processing, and social network analysis.

PENG REN (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in electronic engineering from the Harbin Institute of Technology, Harbin, China, and the Ph.D. degree in computer science from the University of York, York, U.K. He is currently a Professor with the College of Oceanography and Space Informatics, China University of Petroleum, Qingdao, China. His research interests include remote sensing and machine learning. Dr. Ren was a recipient of the K. M. Scott Prize from the University of York, in 2011 and the Eduardo Caianiello Best Student Paper Award at the 18th International Conference on Image Analysis and Processing, in 2015, as one Coauthor.