FPGA-Based Hardware Modeling on Pigeon-Inspired Optimization Algorithm

Yu Zhao, Chun Zhao(✉), and Yue Liu

Beijing Information Science and Technology University,
School of Computer Science, Institution of Smart Manufacturing & Complex System,
BISTU, Beijing 100101, China
zhaochun@bistu.edu.cn

Abstract. By learning behavioral characteristics and biological phenomena in nature, such as birds, ants, and fireflies, intelligent optimization algorithms (IOA) is proposed. IOA shows feasibility in solving complex optimization problems in reality. Pigeon-inspired optimization (PIO) algorithm, which belongs to intelligent optimization algorithms, is proposed by the pigeons homing navigation behavior inspired. PIO is superior to other algorithms in dealing with many optimization problems. However, the performance of PIO processing large-scale complex optimization problems is poor and the execution time is long. Population-based optimization algorithms (such as PIO) can be optimized by parallel processing, which enables PIO to be implemented in hardware for improving execution times. This paper proposes a hardware modeling method of PIO based on FPGA. The method focuses on the parallelism of multi-individuals and multi-dimensions in pigeon population. For further acceleration, this work uses parallel bubble sort algorithm and multiply-and-accumulator (MAC) pipeline design. The simulation result shows that the implementation of PIO based on FPGA can effectively improve the computing capability of PIO and deal with complex practical problems.

Keywords: Intelligent optimization algorithm · Pigeon-inspired optimization · FPGA

1 Introduction

Intelligent optimization algorithm (IOA), also known as meta-heuristic algorithm, is suitable for parallel processing, with strong versatility, good robustness. IOA gets good results in a wide range of scientific and practical problems. In recent years, IOA attracts a lot of research interest worldwide. Li et al. propose a simulation design optimization method of IOA based on MATLAB, which

Supported by the National Key R&D Program of China (No. 2018YFB1701600).

effectively shorten the simulation design cycle and improve the accuracy of the design to a certain extent [1]. Li et al. propose an extended Kalman filter (EKF) algorithm based on particle swarm optimization (PSO), which can get smaller filtering deviation for aircraft trajectory tracking [2].

Inspired by the homing behavior of pigeons, Duan and Qiao propose pigeon-inspired optimization (PIO) in 2014 [3]. PIO shows good performance and receives extensive research. Typically, Duan and Qiu propose a multi-objective PIO method, which is suitable for solving multi-objective optimization problems and successfully applies to parameter design of brushless direct current motor [4]. Alazzam et al. propose a discrete pigeon-inspired optimizer to solve the multiple traveling salesmen Problem [5]. Zhang and Duan propose a modified PIO model adopting Gaussian strategy and verify the feasibility and effectiveness in solving orbital spacecraft formation reconfiguration problems [6]. Moreover, PIO is also used to solve other problems, such as parameter design of the controller for small unmanned helicopters [7], fuzzy energy management strategy for parallel hybrid electric vehicle [8], Underwater Wireless Sensor Networks [9], the target detection task for Unmanned Aerial Vehicles (UAVs) at low altitude [10], feature selection by improving binary pigeon-inspired optimization [11], active disturbance rejection attitude controller for quadrotors [12] etc.

However, there are complex scenarios, large scale and complex computing problems in the real world, such as multi-task assignment, mixed resource scheduling, complex control parameters. In application scenarios dealing with large-scale and complex problems, the IOA, including PIO, execution efficiency decreases. As the calculation complexity increases, the execution time of the PIO increases rapidly, which degrades the performance of PIO. Due to the inherent parallelism of the PIO, PIO can be designed on parallel acceleration platforms to solve the above problems. Currently, studies on parallel design of IOA are mainly divided into the following three patterns: multi-core (open multi-processing), distributed (MapReduce), and heterogeneous computing-based parallel platforms (graphics processing unit-GPU, FPGA, Application Specific Integrated Circuit-ASIC). The first two methods take a general-purpose computer as the hardware carrier and divide tasks into several parts for parallel execution, but still serial in nature. Heterogeneous computing refers to the use of hardware with different architectures to speed up the task of computing. GPU as a parallel acceleration platform is widely used, but GPU has the disadvantages of high power consumption and insufficient flexibility. Using AISC to accelerates IOA algorithm is also feasible, but ASIC is expensive and Not reconfigurable. FPGA is a reconfigurable chip with flexibility and low energy consumption. Parallel design technology can effectively accelerate IOA on FPGA platform. therefore, this paper presents a PIO hardware modeling method based on FPGA, including parallelization analysis, state machine design, and IP core scheduling.

The rest of the paper is structured as follows. Section 2 reviews the typical hardware modeling and implementation of IOA. Section 3 introduces the proposed method. Result and analysis are carried out in Sect. 4. Section 5 concludes this paper.

2 Related Work

Intelligent optimization algorithms encounter challenges in the face of complex calculations. In order to solve the above problems, researchers design various acceleration methods on various platforms. There are some studies on parallel design based on classic algorithms such as genetic algorithm (GA), particle swarm optimization (PSO) and ant colony algorithm (ACO), while few research on PIO. Zou et al. accelerate GA and PSO algorithm by using FPGA, open multi-processing and Compute Unified Device Architecture (CUDA) [13]. Zhou et al. execute PSO in parallel on GPU by using the general-purpose computing ability of GPU and based on the software platform of CUDA from NVIDIA [14]. Menezes et al. propose a parallel design for ACO GPU-based. in which distinct parallelization strategies are compared and analyzed [15]. Juang et al. propose a parallel design of ACO and apply to a fuzzy controller [16]. Moreover, Djenouri et al. improve bees swarm optimization algorithm by using GPU parallelism [17]. Jiang et al. accelerate Whale Optimization Algorithm (WOA) by using OpenCL-based FPGA parallel design [18]. Sadeeq et al. implement the firefly optimization algorithm based on FPGA [19].

3 Hardware Modeling of PIO Based on FPGA

This section introduces the hardware modeling of PIO, first introduces briefly PIO, then conducts modeling analysis from the perspective of acceleration, and finally the detailed design method is given.

3.1 Pigeon-Inspired Optimization Algorithm

The PIO is proposed by mimicking the special navigation behavior of pigeons in the homing process. In PIO, two operators are designed to mimic the mechanism which pigeons use different navigation tools at different stages of finding a target.

Map and Compass Operator
Pigeons can sense the earth's magnetic field using magnetic objects and form a mental map. Pigeons use the sun's altitude as a compass to steer flight, and as pigeons approach target, pigeons rely less on the sun and magnetic field.

Landmark Operator
The landmark operator is used to mimics the influence of landmarks on pigeons in navigation tools. The closer the pigeons get to destination, the more the pigeons rely on landmarks. Pigeons familiar with landmarks fly straight to destination. And pigeons unfamiliar with the landmark follow pigeons familiar with the landmark.

Suppose the search space is n-dimensional, the i-th pigeon can be represented by an n-dimensional vector, $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,n})$. The velocity of the pigeon, which represents the change in position of the pigeons, can be represented by another n-dimensional vector, $V_i = (v_{i,1}, v_{i,2}, ..., v_{i,n})$. The global best position obtained by comparing the positions of all pigeons after each iteration is $X_g = (x_{g,1}, x_{g,2}, ..., x_{g,n})$. Then, each pigeon updates velocity and position according to the following two equations:

$$V_i(t) = V_i(t-1)e^{-Rt} + rand(X_g - X_i(t-1)) \tag{1}$$

$$X_i(t) = X_i(t-1) + V_i(t) \tag{2}$$

where t is the current number of iterations; R is the map and compass factor, with a range of $[0, 1)$, which is used to control the impact of the latest velocity on the current velocity; $rand$ is a random number, uniformly distributed in $[0, 1)$; Eq. (1) is used to update velocity of the pigeon according to latest velocity of the pigeon and the distance of current position of the pigeon from the global best position. Then the pigeon updates position with new velocity of the pigeon according to Eq. (2). As the number of iterations reaches the requirement, stop the work of map and compass operator and continue to work in landmark operator.

In the landmark operator, pigeons depend on landmarks for flight. After each iteration, the number of pigeons decrease by half by Eq. 3. Pigeons far from destination are unfamiliar with the landmarks and can not discern the path, such pigeons are discarded. X_c is the center position of the remaining pigeons, which be used as a landmark and reference for flying. The equation in the landmark operator is given as follow:

$$N_p(t) = \frac{N_p(t-1)}{2} \tag{3}$$

$$X_c(t) = \frac{\sum_{n=1}^{N_p(t)} X_i(t) fitness(X_i(t))}{N_p \sum_{n=1}^{N_p(t)} fitness(X_i(t))} \tag{4}$$

$$X_i(t) = X_i(t-1) + rand(X_c(t) - X_i(t-1)) \tag{5}$$

where N_p is the size of the population; $fitness$ is an evaluation function calculating the fitness of each pigeon. Equation (4) is used to calculate the center value of pigeons. Then the pigeon flies toward a new position according to Eq. (5). As the number of iterations of the landmark operator reaches the requirement, the landmark operator stops working and the algorithm is finished.

3.2 Hardware Modeling Analysis of PIO

In order to improve the execution efficiency of the algorithm, finding the acceleration in hardware modeling is necessary. Four kinds of acceleration are proposed as follows.

Multi-individual Parallelism
The intelligent optimization algorithm based on population have good parallelism because the individual in each iteration is independent from each other. Specifically, velocity updates, position updates, and individual evaluations of PIO can all be processed in parallel.

Multi-dimensional Parallelism
Inside each operation of PIO, the dimensions are independent of each other. Thus, multi-dimensional parallel operations can be implemented. Assuming a solution space of 10 dimensions, the speed of multidimensional operations can theoretically be increased by nearly 10 times.

Pipeline Design of Multiply-and-Accumulator (MAC) Circuit
To calculate the center value of the pigeons by Eq. (4), MAC operation is required. This work can carry out pipeline design of MAC (see Fig. 1) to accelerate PIO. Pipeline design enables adder and multiplier to be fully utilized in limited time.

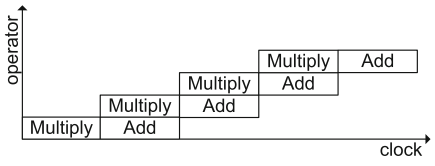

Fig. 1. Pipeline design of MAC circuit.

Sort Algorithm Parallelism
In the landmark operator, sorting is required before calculate the center value of better pigeons. This work exploits parallelism to implement bubble sort algorithm. Parallel bubble sort allows parallel comparison and swapping of independent data [20]. The Odd-even transposition technique is well suited for this case. Let

$$A =< a_1, a_2, a_3, ..., a_i, a_j, ..., a_n > \tag{6}$$

is a list of n elements. Any pair of adjacent elements is called an element pair, such as a_i, a_j. If the a_i is odd, the element pair is an odd pair, and if the a_i is even, the element pair is an even pair.

The descending order of 10 numbers is taken as an example to illustrate parallel bubble sort. A basic operator (see Fig. 2) including a phase of odd pair and a phase of even pair which work sequentially is defined. But each phase is compared and swapped in parallel. The parallel bubble sort algorithm is given below (see Algorithm. 1).

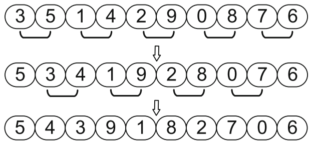

Fig. 2. Base operation of parallel bubble sort.

3.3 Parallel Design of PIO Based on FPGA

A new FPGA can be treated as a blank processor. Except specific logical resources and other auxiliary computing resources, the hardware circuit can be automatically generated after the HDL code is downloaded to the FPGA chip.

In this work, the pigeon population is stored in the top-level module's signal. And the PIO parameters are declared as follows: solution space dimension $D = 10$, the population size $N_p = 10$, map and compass factor $R = 0.2$, the number of iteration $iter1 = 15$ and $iter2 = 15$ for two operators.

The program flow design of PIO based on FPGA is shown in Fig. 3. And the data flow is shown in Fig. 4.

Control Unit Design in FPGA
Since there is no controller in FPGA, designing the control unit (CU) is necessary. CU is responsible for state switching within the algorithm, such as timing control, communication, reading and writing memory, etc. There is also a CU inside each module to improve the parallelism of the algorithm. In the top-level module design, multiple Arithmetic logical unit (ALU) modules are used to carry out multi-individual parallelism through CU.

Arithmetic Logical Unit (ALU) Design in FPGA
There is no fixed computing architecture and ALU in FPGA. Considering the mathematical formula of PIO, designing ALU is necessary. Since there is no decimal point calculation in FPGA, the design of ALU in this work chooses floating point operation in accordance with IEEE754 international standard. ALU can be used in parallel if there are sufficient logical resources in the FPGA.

Ten ALU modules are designed in this work, which are evaluation module, sorting module, updating best fitness module, exponent module, random module, updating velocity module in compass operator, updating position module in compass operator, MAC module, center value module, and updating position module in landmark operator. To make the description clearer, some control signals, such as reset signal and enable signal, are hidden and each module is described as follows:

Algorithm 1. Parallel bubble sort algorithm (descending sort)

```
Input: A(an unsorted array of length n)
Output: A(an sorted array of length n)
  $i \leftarrow 0$
  while $i < \frac{n}{2}$ do
      $j \leftarrow 0$
      while $j < n - 1$ do                                  /* Do in parallel */
          if $A[j] < A[j+1]$ then
              $temp \leftarrow A[j]$
              $A[j] \leftarrow A[j+1]$
              $A[j+1] \leftarrow temp$
          end if
          $j \leftarrow j + 2$
      end while
      $j \leftarrow 1$
      while $j < n - 1$ do                                  /* Do in parallel */
          if $A[j] < A[j+1]$ then
              $temp \leftarrow A[j]$
              $A[j] \leftarrow A[j+1]$
              $A[j+1] \leftarrow temp$
          end if
          $j \leftarrow j + 2$
      end while
      $i = i + 1$
  end while
```

Evaluation Module

In this work, using a general benchmark function [21] as fitness function to carry out hardware modeling of PIO based on FPGA, and the benchmark function is:

$$fitness(X_i) = \sum_{i=1}^{n} [x_i + 0.5]^2 \tag{7}$$

where X_i is the i-th pigeon, x_i is i-th dimension of the pigeon. By comparative experiments, PIO gets good performance while x_i is $[0, 15]$. Hardware modeling of evaluation module is shown in Fig. 5. The module operates according to Eq. (7). The module uses multi-dimensional parallelism and receives a position and then outputs a fitness value.

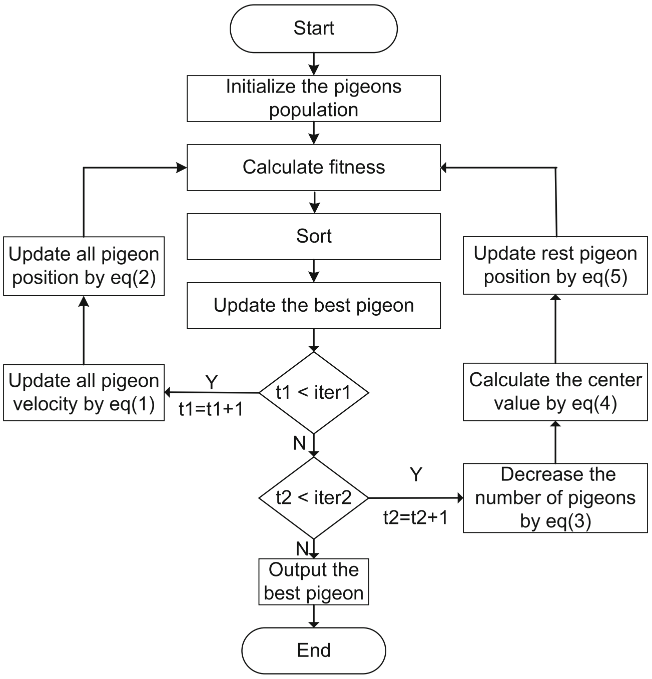
Start
Initialize the pigeons population
Calculate fitness
Sort
Update the best pigeon
t1 < iter1
Y
t1=t1+1
Update all pigeon velocity by eq(1)
Update all pigeon position by eq(2)
N
t2 < iter2
Y
t2=t2+1
Decrease the number of pigeons by eq(3)
Calculate the center value by eq(4)
Update rest pigeon position by eq(5)
N
Output the best pigeon
End

Fig. 3. Program flow design of PIO.

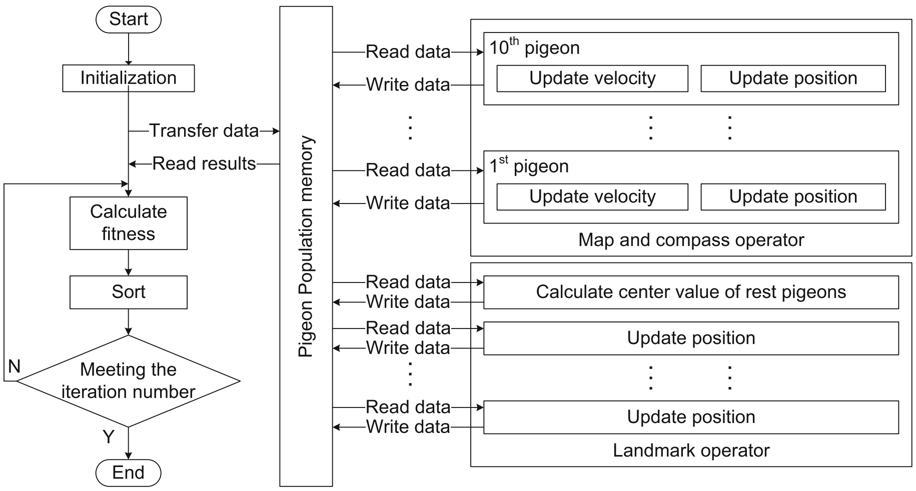

Fig. 4. Data flow design of PIO.

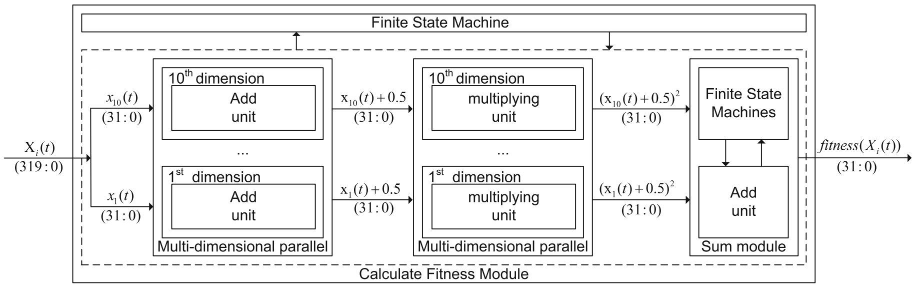

Fig. 5. Hardware modeling of evaluation.

Sorting Module
The sorting module sorts the pigeons according to fitness by using the above parallel bubble sorting algorithm, and the hardware modeling of sorting is shown in Fig. 6. The swapped module in the figure is used to compare and swap two pigeons. Considering module reuse, the sorting module uses only five swapped modules. More swapped modules are shown in the figure for ease of description.

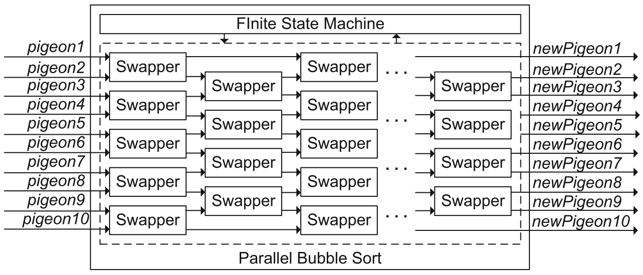

Fig. 6. Hardware modeling of Sorting.

Updating Best Fitness Module
After the sorting is finished, by comparing two fitness values, the X_g may be updated. Hardware modeling of updating best fitness module is shown in Fig. 7.

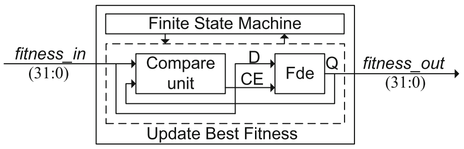

Fig. 7. Hardware modeling of updating best fitness

Exponent Module
Approximation value of e^x can be got by the kth-order Taylor polynomial (see Eq. (8)). The accuracy of e^x module changes as k changes (see Fig. 8). By contrast experiment, let k = 5 can get sufficient precision. Hardware modeling of exponent module is shown in Fig. 9.

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} \tag{8}$$

Random Module
This work design random number generator by linear Feedback Shift Registers (LFSR) [22]. A 8-bit random binary sequences can be got by 8-bit LFSR (see Fig. 10). To get the random number, uniformly distributed in [0, 1), the “8-bit random binary sequences” is defined as a 9-bit fixed point number. The fixed point number consists of 1-bit sign digits, 3-bit integer digits and 5-bit fractional digits. The value of 1-bit sign digits is 0. Obviously, the ranges of fixed point number is [0, 8). And then the random number uniformly distributed in [0, 1) can be got by divide the fixed point number by eight.

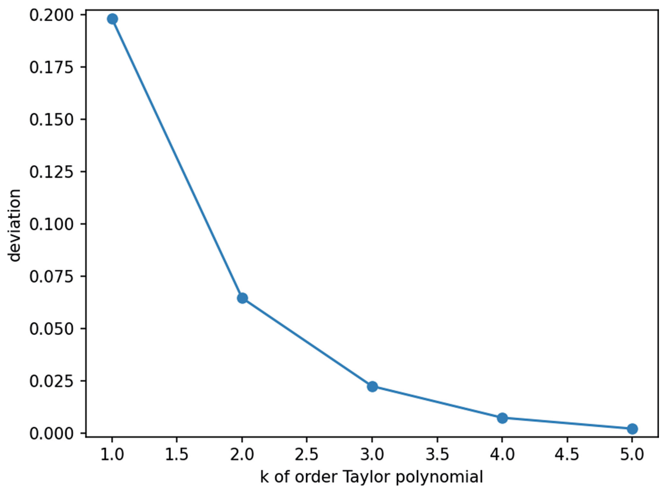

Fig. 8. Precision of exponent module,which x in range [0, 15].

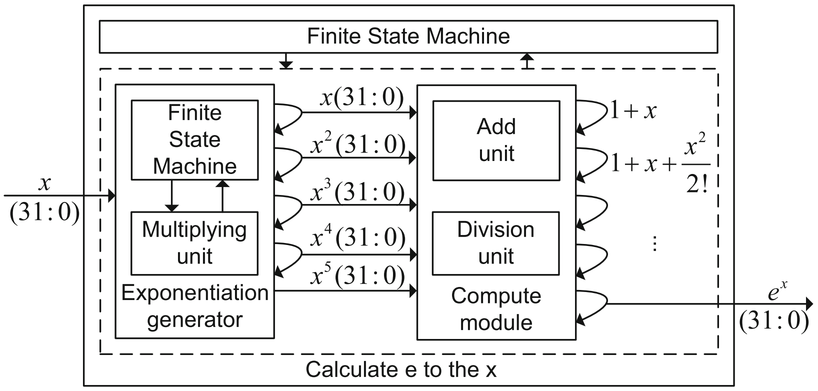

Fig. 9. Hardware modeling of exponent.

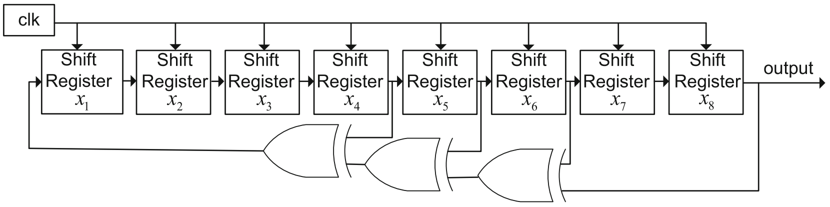

Fig. 10. 8-bit LFSR design.

Updating Velocity Module in Compass Operator
The updating velocity module in compass operator is used to update the velocity of pigeon by Eq. (1). Hardware modeling of updating velocity module in compass operator is shown in Fig. 11, which uses multi-dimensional parallelism.

Updating Position Module in Compass operator
The updating position module in compass operator is used to update the position of pigeon by Eq. (2). Hardware modeling of updating position module in compass operator is shown in Fig. 12, which uses multi-dimensional parallelism.

MAC Module
To calculate X_c by Eq. (4), MAC circuit is required. Hardware modeling of MAC is shown in Fig. 13. Through the control of the finite state machine, the module uses pipeline design.

Center Value Module
In the landmark operator, pigeons of low fitness value are discarded by Eq. (3), and then the center value of remaining pigeons are calculated. To get the center value (X_c), a center value module using multi-dimensional parallelism is designed(see Fig. 14).

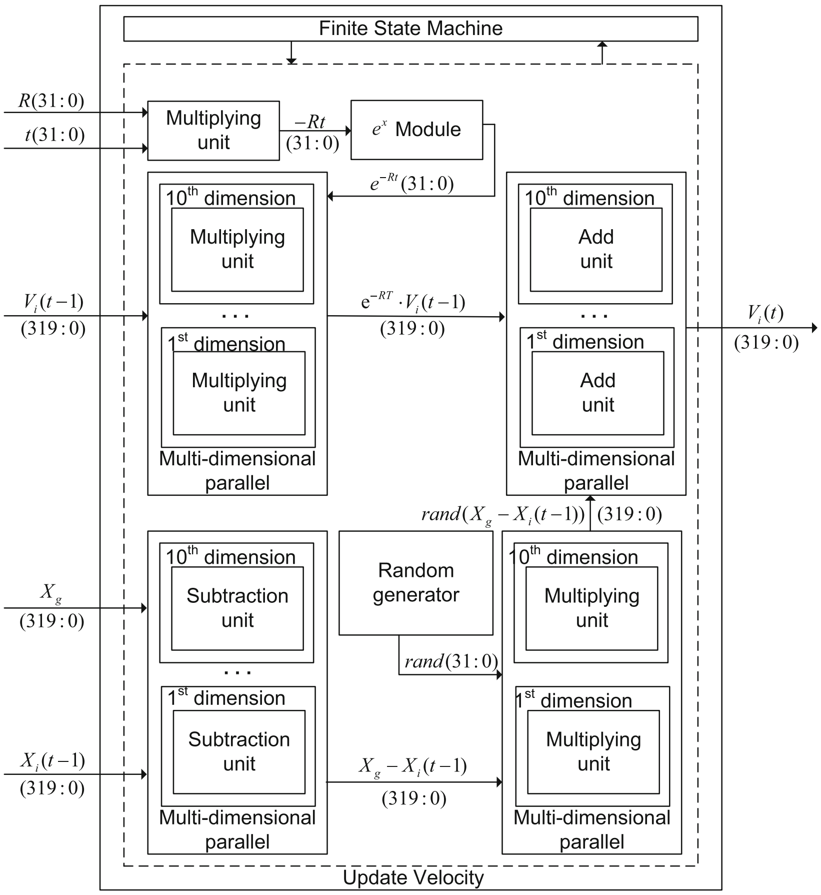

Fig. 11. Hardware modeling of updating velocity.

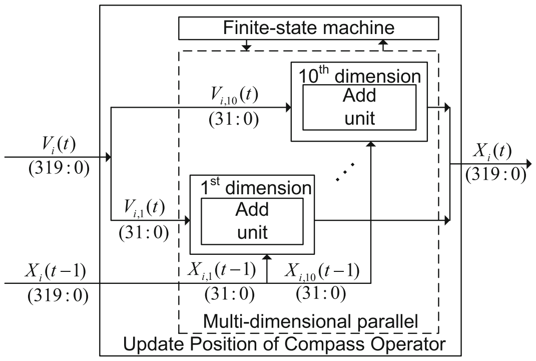

Fig. 12. Hardware modeling of updating position in the compass operator.

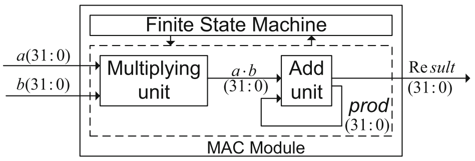

Fig. 13. Hardware modeling of MAC

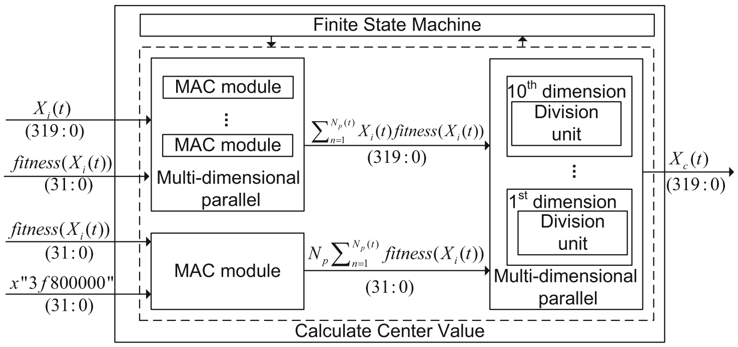

Fig. 14. Hardware modeling of center value

Updating Position module in landmark operator
The updating position module in landmark operator is used to update the position of pigeon by Eq. (5). Hardware modeling of updating position module in landmark operator is shown in Fig. 15, which uses multi-dimensional parallelism.

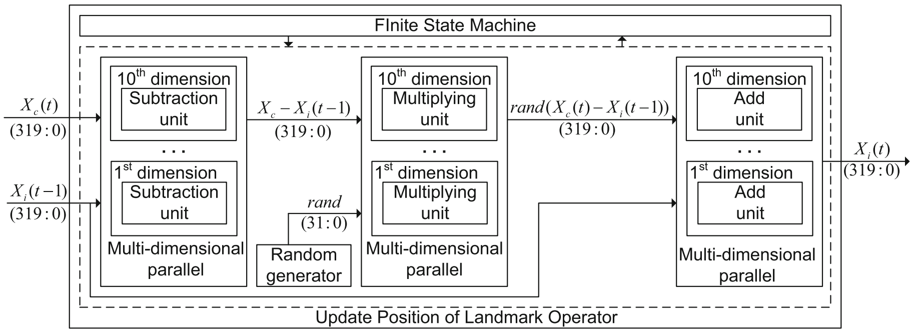
FInite State Machine
$X_c(t)$
$(319:0)$
$X_i(t-1)$
$(319:0)$
10th dimension Subtraction unit
1st dimension Subtraction unit
Multi-dimensional parallel
$X_c - X_i(t-1)$
$(319:0)$
$rand$
$(31:0)$
Random generator
10th dimension Multiplying unit
1st dimension Multiplying unit
Multi-dimensional parallel
$rand(X_c(t) - X_i(t-1))$
$(319:0)$
10th dimension Add unit
1st dimension Add unit
Multi-dimensional parallel
$X_i(t)$
$(319:0)$
Update Position of Landmark Operator

Fig. 15. Hardware modeling of updating position in the landmark operator

4 Results and Analysis

Equation (7) is the sum of squares formula, and x_i in range $[0, 15]$. Obviously, fitness get minimum value 2.5 while $x_i = 0, i = 1, 2, 3, ..., 10$.

This work codes VHDL in ISE14.7 and gets the simulation waveform (see Fig. 16). The best fitness of the pigeon population can be obtained after initialization is 562.14. After the compass and map operator, the best fitness of pigeon population is reduced to 452.41. After the landmark operator, the best fitness value reduces rapidly to 2.54, approximately reaching the true best value 2.5. In addition, clocks cost of each module is shown in Table 1. For comparison, Table 2 shows the cost of clocks required for each numerical operation. The numbers involved in the numerical operation are all IEEE754 single-precision floating-point format.

The exponent of e module takes a lot of clocks because the exponent of e module requires massive multiplication, division, and addition by Eq. (8), which causes the updating velocity module in compass operator also takes a lot of clocks. The MAC module finishes a multiplication and addition only need 16 clock. The center value module requires ten times multiplication and addition, so the center value module also needs a lot of clocks. By comparison, parallel bubble sort is 6.95 times faster than serial bubble sort, and the more the number of sorts, the more obvious the acceleration effect.

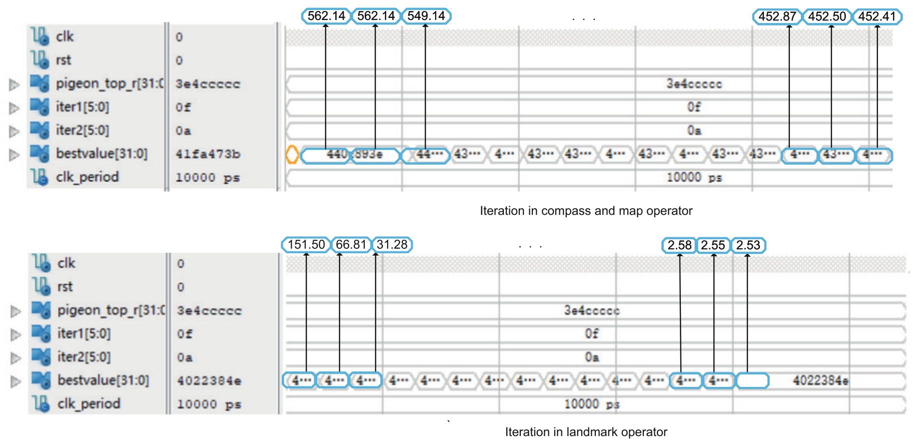

Fig. 16. Simulation waveform

Table 1. Clock cost of each module.

Module name	Clocks
Evaluation module	93
Sorting module	148
Updating best fitness module	4
Exponent module	259
Random module	43
Updating velocity module in compass operator	370
Updating position module in compass operator	34
MAC module	16
Center value module	242
Updating position module in landmark operator	88

Table 2. Clock cost of each numeric operation.

Operation	Clocks
Add	12
Subtraction	12
Multiplication	8
Division	28

5 Conclusion

This paper presents a method of hardware modeling of PIO based on FPGA, and improve the compute performance by parallel design. Firstly, modeling analysis is carried out from the perspective of acceleration, and then the program flow design and data flow design are presented. In addition, the research is a reference for hardware modeling of other intelligent optimization algorithms.

In this work, state machine control is used in the whole process of hardware modeling, including timing control, communication, reading and writing memory, etc. The ALU module uses multi-dimensional parallelism to accelerate PIO. Through CU, multiple ALUs can be used in parallel to accelerate PIO.

In the future, parallel design of PIO can map to a specific FPGA chip to validate PIO performance by comparing with other implementation methods. In addition, PIO based on FPGA can be applied to practical engineering problems.

Acknowledgment. This work is supported by the National Key R&D Program of China (No. 2018YFB1701600).

References

1. Li, W., et al.: A simulation design and optimization method based on MATLAB and intelligent optimization algorithm. In: Proceedings of 2020 China Simulation Conference, pp. 396–402 (2020)
2. Li, L., et al.: Improved EKF aircraft trajectory tracking algorithm based on PSO. In: Proceedings of the 33rd China Simulation Conference, pp. 64–69 (2021)
3. Duan, H., Qiao, P.: Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. Int. J. Intell. Comput. Cybern. **7**(1), 24–37 (2014)
4. Qiu, H.X., Duan, H.B.: Multi-objective pigeon-inspired optimization for brushless direct current motor parameter design. Sci. China Technol. Sci. **58**(11), 1915–1923 (2015). https://doi.org/10.1007/s11431-015-5860-x
5. Alazzam, H., Alsmady, A., Mardini, W.: Solving multiple traveling salesmen problem using discrete pigeon inspired optimizer. In: 2020 11th International Conference on Information and Communication Systems (ICICS). IEEE (2020)
6. Zhang, S., Duan, H.: Gaussian pigeon-inspired optimization approach to orbital spacecraft formation reconfiguration. Chin. J. Aeronaut. **28**(1), 200–205 (2015)
7. Zhang, D., Duan, H., Yang,Y.: Active disturbance rejection control for small unmanned helicopters via Levy flight-based pigeon-inspired optimization. Aircraft Engineering and Aerospace Technology (2017)
8. Pei, J.Z., YiXin, S., Zhang, D.H.: Fuzzy energy management strategy for parallel HEV based on pigeon-inspired optimization algorithm. Sci. China Technol. Sci. **60**(3), 425–433 (2017)
9. Yu, S., et al.: Node self-deployment algorithm based on pigeon swarm optimization for underwater wireless sensor networks. Sensors **17**(4), 674 (2017)
10. Li, C., Duan, H.: Target detection approach for UAVs via improved pigeon-inspired optimization and edge potential function. Aerosp. Sci. Technol. **39**, 352–360 (2014)
11. Pan, J.-S., et al.: Improved binary pigeon-inspired optimization and its application for feature selection. Appl. Intell. **51**(12), 8661–8679 (2021)
12. Yuan, Y., Duan, H.: Active disturbance rejection attitude control of unmanned quadrotor via paired coevolution pigeon-inspired optimization. Aircraft Engineering and Aerospace Technology (2021)
13. Zou, X., et al.: Parallel design of intelligent optimization algorithm based on FPGA. Int. J. Adv. Manuf. Technol. **94**(9), 3399–3412 (2018)
14. Zhou, Y., Tan, Y.: GPU-based parallel particle swarm optimization. In: 2009 IEEE Congress on Evolutionary Computation. IEEE (2009)
15. Menezes, B.A.M., et al.: Parallelization strategies for GPU-based ant colony optimization solving the traveling salesman problem. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE (2019)
16. Juang, C.-F., et al.: Ant colony optimization algorithm for fuzzy controller design and its FPGA implementation. IEEE Trans. Ind. Electron. **55**(3), 1453–1462 (2008)
17. Djenouri, Y., et al.: Exploiting GPU parallelism in improving bees swarm optimization for mining big transactional databases. Inf. Sci. **496**, 326–342 (2019)
18. Jiang, Q., et al.: Improving the performance of whale optimization algorithm through OpenCL-based FPGA accelerator. In: Complexity 2020 (2020)
19. Sadeeq, H., Abdulazeez, A.M.: Hardware implementation of firefly optimization algorithm using FPGAs. In: 2018 International Conference on Advanced Science and Engineering (ICOASE). IEEE (2018)

20. Lipu, A.R., et al.: Exploiting parallelism for faster implementation of Bubble sort algorithm using FPGA. In: 2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE). IEEE (2016)
21. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)
22. Babitha, P.K., Thushara, T., Dechakka, M.P.: FPGA based N-bit LFSR to generate random sequence number. Int. J. Eng. Res. General Sci. **3**(3), 6–10 (2015)