Design of FPGA Hardware based on Genetic Algorithm

Xinxin Sun [a], Juan Li [b], Fenxian Tian [c], Yanshuang Chen [d] and Jun Yang [e, *]

School of Information Science and Engineering, Yunnan University Yunnan Kunming 650091, China

[a] 1015462008@qq.com, [b]1561437500@qq.com, [c]1878777810@qq.com, [d]yschen@163.com

[e, *] junyang@ynu.edu.cn

Abstract. In order to improve the running speed of hardware and the utilization rate of resources, firstly, this paper use genetic algorithm to realize serial and pipeline hardware implementation. Then, the genetic algorithm is improved on the implementation method. In the process of improvement, the pipeline is introduced into the parallel mechanism. Finally, the TSP problem and function extreme value problem are used to verify the resource consumption and running speed of FPGA respectively. The experimental results in this paper show that the two schemes are less expensive in terms of hardware implementation, less expensive in operation and high in efficiency, and the algorithm can also be widely used in the occasions with higher applicability.

Keywords: FPGA; Genetic algorithm; Parallelism; Pipeline structure.

1. Introduction

Genetic algorithm is a highly parallel, random and adaptive search algorithm developed on the basis of mimicking the natural selection and evolution mechanism of natural organisms[1]. After several decades of rapid development, genetic algorithm in combination optimization, pattern recognition, image processing, artificial intelligence, machine learning and etc. has been widely used. FPGA has a regular internal logic array and rich networking resources, the emergence of a new generation of field programmable gate array (FPGA) and the use of hardware description language Verilog HDL enable genetic algorithm to be realized by hardware[2]. This paper gives the basic structure of each module implemented by hardware, designs the serial and pipeline structure of the algorithm, and effectively introduces the parallel mechanism. The program used in this paper uses Altera's EP2C35F672C6N chip, using Verilog HDL design language, and selected the Quarters II 13.0 platform programming.

2. Algorithm Structure Analysis

Genetic algorithm is a multi-point parallel iterative search algorithm[3]. The population diagram of genetic algorithm is shown in Fig.1.

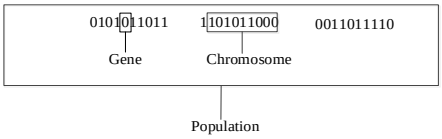

Figure 1. Population diagram of genetic algorithm

The idea of genetic algorithm is inspired by the natural selection and genetic mechanism in the biological world [4]. In the process of realization, it draws on the natural selection and the mutation and mating in the genetic process. Genetic operators are mainly composed of selection, crossover and mutation operators. The operation process of the genetic algorithm is shown in Fig. 2.

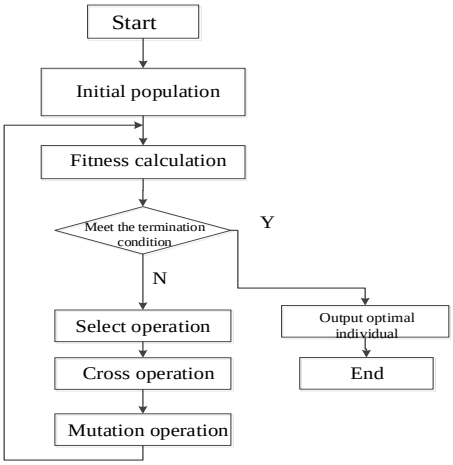

Figure 2. Flow chart of genetic algorithm

Coding, genetic operators and adaptive function calculations are described below.

2.1 Individual Coding

Genetic algorithm can solve some complex problems, such as TSP problems and function extreme problems, individual codes in TSP and function extreme problems use binary coding[5]. Assuming that N cities are needed to solve the TSP problem, and the serial number of each city is represented by binary code, each individual can be expressed as:

$D_{n-1, m-1}D_{n-1, m-2}, \ldots, D_{n-1,0}, D_{n-2, m-1}D_{n-2, m-2}, \ldots, D_{n-2,0}, \ldots, D_{0, m-1}D_{0, m-2}, \ldots, D_{0,0}$

The total length of the individuals can be expressed as M by N. In order to facilitate the description of the problem, assuming the number of cities N=8, then the binary digit number of each city can be obtained as M=3. The following is the solution function for the extremum of the function: $f = {x_1}^2 + {x_2}^2, 0 \leq x_1, x_2 \leq 255$

2.2 Genetic Operator

The key of genetic algorithm is genetic operator, which can be defined as follows in the process of hardware implementation in this paper.

Selection Operator: Due to the convenience of FPGA hardware in the implementation process, this paper uses the random league selection method.

Crossover operator: In this paper, double-point crossover is more suitable than single-point crossover, so double-point crossover is selected.

Mutation operator: Mutation operator generally adopts the selection bit variation. The specific number of selection bit variation is determined according to the individual length, and the selection criterion is that it does not produce much damage.

2.3 Fitness Calculation

One of the important indicators affecting the acceleration of genetic algorithms is the adaptive function calculation.

The adaptation function of the function extremum is: $f = {x_1}^2 + {x_2}^2, 0 \leq x_1, x_2 \leq 255$.

The adaptation function selected by TSP is: $\min\ l = \sigma d(t_i, t_{i+1})(i = 0,1, \cdots, n-1)$ n = 8.

According to the adaptation function selected in the above TSP problem, each city can be represented by a 3-digit binary code, and the travel plan can be a kind of city starting from (t0, t1, t i, …, t n - 1) arrangement. Σd (t i, t i + 1) (i=0,1, …, n-1) represents the cost of a travel plan.

3. Hardware Structure

3.1 The Overall Structure of the Hardware

For simple genetic algorithms, the traditional implementation is serial [6], but In Fig 3, the serial genetic algorithm implementation design does not implement parallel operation. In order to solve this kind of problem, the design of each module is carried out according to the pipeline mode, and adds parallel mechanism on the basis of pipeline, as shown in Fig. 4.

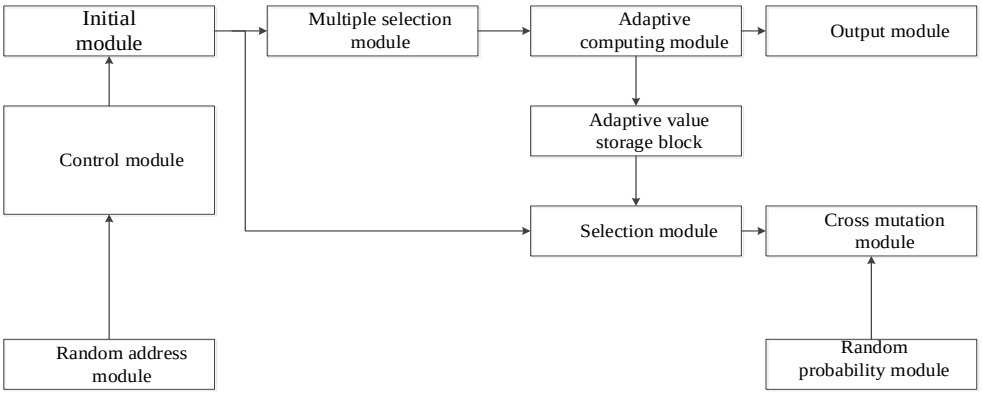

Figure 3.design of serial genetic algorithm

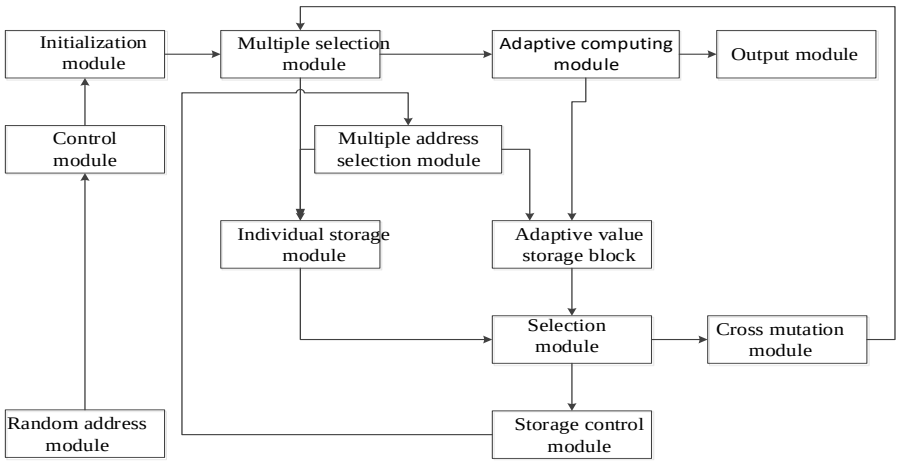

Figure 4. Flow water genetic algorithm

In this paper, pipeline mechanism and parallel mechanism are added to the improved genetic algorithm. The improved system has a total of 12 functional modules. And each module in parallel will greatly save running time and evolution time, while also reducing the use of chip data.

3.2 System Main Function Module Design

3.2.1 Selection Module

The selection module performs a selection operation in the genetic algorithm[7]. This article will use random league competition to select individuals. Whenever the start signal of the selection module gradually becomes effective, the module will start to work normally, and each time two groups of individuals will be read. First, the two groups will be selected twice, and the individuals with larger adaptation values in each group will be selected for output. The state machine is shown in Fig.5.

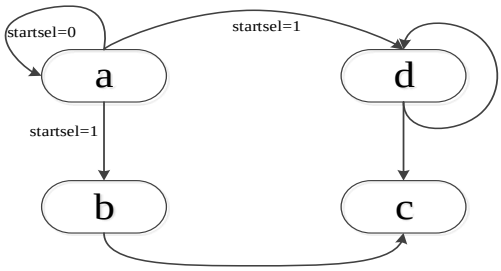

Figure 5. Select module state machine

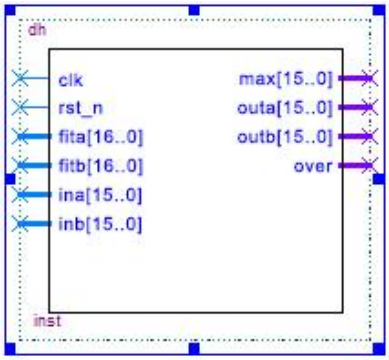

Figure 6. Select module port design

Fig.6 is the design drawing of the selected module port, and the simulation is shown in Fig.7.

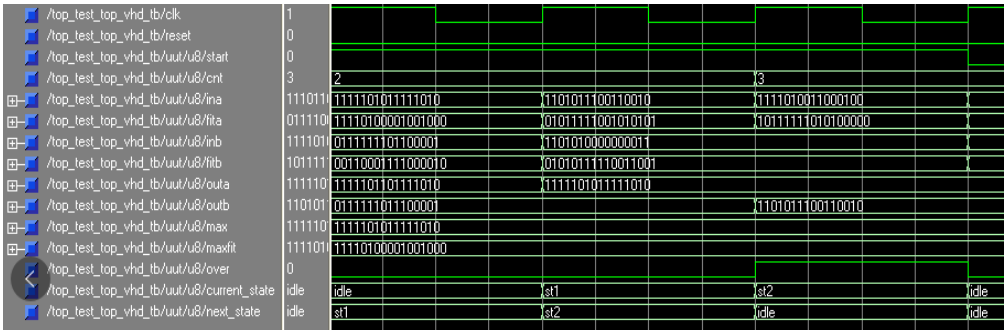

Figure 7. Select module simulation diagram

Fig.7 is a simulation result of the system when it is executed. It can be seen from the figure that the output port of the selection module will maintain the previous value, and the max signal in the figure represents the maximum value of the fitness function of a generation. The experimental results show that the simulation results are consistent with the actual calculation results.

3.2.2 Control Module

The control module is the core part of the whole design, which controls the flow of the whole design and the flow of data. Each module works orderly on the basis of the control module. The control module can also be represented by a state machine. And Fig.8 is a control module state machine.

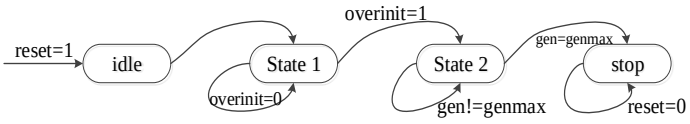

Figure 8. Control Module State Machine

The control module as a whole contains four states. Idle is the reset state, stop is the end state, and both state 1 and state 2 are working states. The transition of these four working states can be obtained by the control module.

Fig.9 is the design drawing of the control module port, and the simulation is shown in Fig.10.

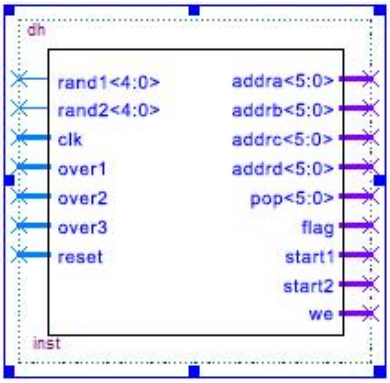

Figure 9. Control module port design

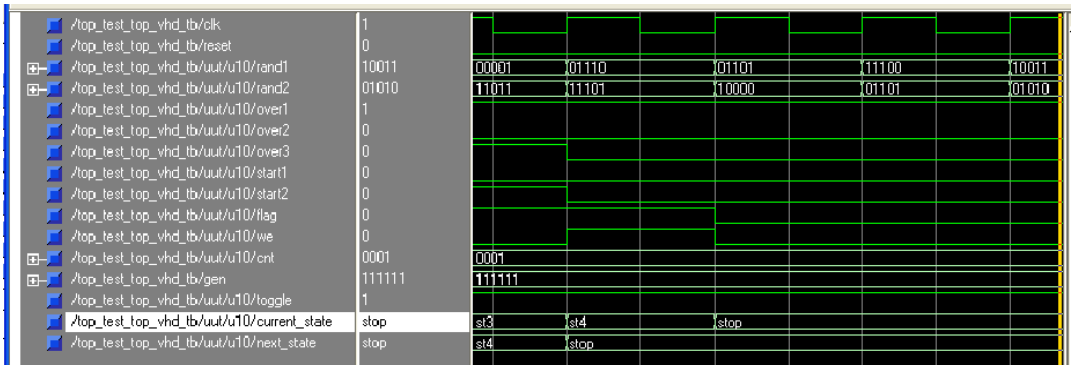

Figure 10. Control module simulation diagram

In the simulation diagram of the control module in Fig 10, when the evolutionary algebra gen changes from "111111" to "000000", the state of current_state will remain unchanged. In this case, we=0 is expressed as read, and start1=0 and start2=0. In other words, the selection and crossover mutation module is in idle state.

3.2.3 Storage Control Module

Fig.11 shows the state machine of the storage control module. Idle is in idle state. When the start signal is started, the module will start immediately and the system enters a state. In a state, the module gives the memory address and write control signal, which are output to multiple address and read/write signal gate module. When the storage generation is complete, the system reenters state b.

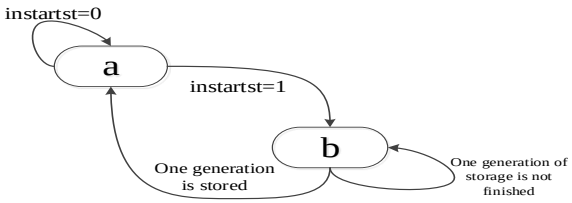

Figure 11. Storage Control Module State Machine

4. Fpga Performance Test based on Genetic Algorithm

The system was designed in Altera's Quartus II 13.0 integrated development environment, simulated in ModelSim 9.0, and finally downloaded to the DE2 development board with EP2C35F672C6N as the main control chip for board level debugging. Fig.12 is a simulation result of the modelsim simulation of the extreme value of the genetic algorithm.

4.1 Experimental Result

According to the experimental data obtained by our experiments and the results of modelsim simulation, when the system starts, there will be two individuals per cycle. Assuming that the parameters we set are consistent, then we use FPGA serial scheme and FPGA flow scheme respectively to solve TSP problem and function extreme value problem. Finally, the following results are compared in the resource consumption of the system to obtain the results as shown in Table 1.

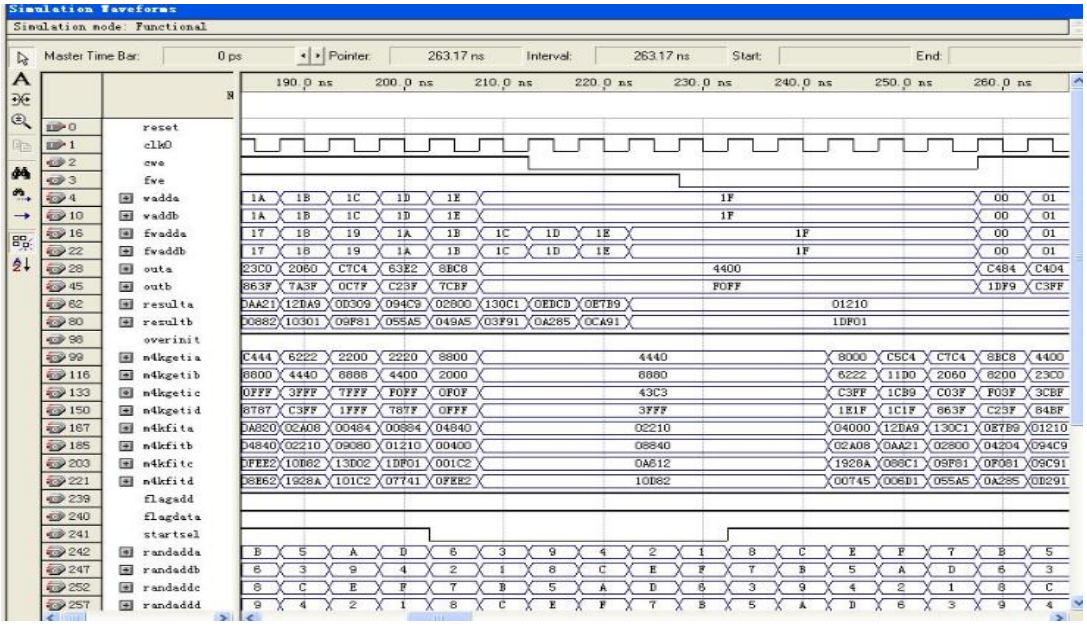

Figure 12. Simulation results of function extremum

Table 1 show the usage and operation schedule of FPAG resources. It can be seen from the data in the table that the flow of water is less than that of serial use in resource use, and the improved flow genetic algorithm can shorten the running time and increase the clock frequency.

Table 1. FPGA resource usage table

Serial number	Resources name	Total resources	Instructions	Extreme value problem	TSP problem
				Number of resources used	Number of resources used
1	Logical unit	66500	FPGA serial GA	680	2006
			FPGA flow water GA	700	2015
2	Storage unit	120KB	FPGA serial GA	240B	496B
			FPGA flow water GA	480B	992B

5. Conclusion

This paper proposes a genetic algorithm for both serial and pipeline solutions. And the hardware structure of the FPGA is adopted. In the implementation of the serial scheme is inefficient, the

pipeline scheme is used. And the parallel mechanism is also introduced in the pipeline, which greatly improves the running speed of the algorithm. In order to verify both serial and pipeline solutions. This paper uses the function extremum and TSP problem to test. The experimental results show that the two schemes are faster than the software in the implementation method. The running time of the algorithm is greatly shortened, and the resource usage is better than the software, which provides the possibility for the practical application of the genetic algorithm.

Acknowledgments

The author, Xinxin Sun, thanks the postgraduate project of Yunnan university school of information for supporting the publication of this paper "research on key technologies of intelligent home robot voice control system based on FFT dual-core processor".

References

[1]. Guo L, Thomas D B, Guo C, et al. Automated framework for FPGA-based parallel genetic algorithms[C]// International Conference on Field Programmable Logic & Applications. 2014.

[2]. Deshpande P U, Bhosale S A. AES encryption engines of many core processor arrays on FPGA by using parallel, pipeline and sequential technique[C]// International Conference on Energy Systems & Applications. 2016.

[3]. Zhong G, Prakash A, Wang S, et al. Design Space exploration of FPGA-based accelerators with multi-level parallelism[C]// Design, Automation & Test in Europe Conference & Exhibition. 2017.

[4]. Lu L, Yun L, Xiao Q, et al. Evaluating Fast Algorithms for Convolutional Neural Networks on FPGAs[C]// IEEE International Symposium on Field-programmable Custom Computing Machines. 2017.

[5]. Li H, Fan X, Li J, et al. A high performance FPGA-based accelerator for large-scale convolutional neural networks[C]// International Conference on Field Programmable Logic & Applications. 2016.

[6]. Zhang Y, Keller N P. Implementation of Genetic Algorithm for TSP Based on FPGA[C]// Proceedings of the 2011 Chinese Control and Decision Conference (CCDC). 2011.

[7]. Chen Y, Wu Q. Design and implementation of PID controller based on FPGA and genetic algorithm[C]// International Conference on Electronics & Optoelectronics. 2011.