SELECTED PAPER AT NCSP'15

Two-Level Pipeline Structure for Particle Swarm Optimization

Ke Fan¹, Kui-Ting Chen², Peikun Wang¹ and Takaaki Baba¹

¹Graduate School of Information, Production and Systems, Waseda University

²Research Center of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0135, Japan

Phone: +81-93-692-5017, FAX: +81-93-692-5021

E-mail: fanker0806@asagi.waseda.jp, nore@akane.waseda.jp, pkwang@fuji.waseda.jp, tbaba@waseda.jp

Abstract

In many nonlinear applications, hardware architectures for the particle swarm optimization (PSO) algorithm are implemented to obtain the optimum solution. However, the conventional hardware implementation is inefficient when the arithmetic equations in the application become complex. To overcome this problem, this paper presents a two-level pipeline structure for the PSO algorithm based on field-programmable gate array (FPGA) technology. The proposed hardware employs two novel features. First, a generic particle calculation block (GPCB) is adopted to support different kinds of PSO algorithm. Second, a two-level pipeline approach is adopted, which helps to increase the calculation speed of the proposed hardware. The experimental results demonstrate that the proposed architecture achieves higher performance than conventional hardware.

1. Introduction

Particle swarm optimization (PSO), originally proposed by Kennedy and Eberhart in 1995 [1], is a stochastic heuristic algorithm based on swarm intelligence. Compared with traditional techniques such as the genetic algorithm (GA), the advantage of the PSO algorithm is attributed to its ability to produce a better result with a higher convergence speed and a lower cost [2, 3]. Therefore, PSO is widely used to obtain the optimal solution in complex optimization problems. However, the PSO algorithm is extremely time-consuming, even when implemented on high-performance computers. To reduce the execution time, PSO hardware implementation is an outstanding approach.

In recent research on the PSO algorithm on hardware, several PSO hardware architectures have been proposed [4-6]. In particular, a pipeline structure for PSO was pre-

sented in [5], which implements serial hardware for PSO with random time-varying inertia weight and acceleration coefficients (PSO-RTVIWAC) [7]. In [5], Cai et al. simplified the existing hardware [8] to reduce the calculation error. Meanwhile a pipeline approach has been used to increase the calculation speed. However, this architecture only supports a particular PSO algorithm, which is not able to meet different requirements in different applications. In addition, the employed pipeline approach only takes the low I/O operation speed of RAM into consideration, without paying enough attention to the complex calculation of particle fitness values.

In this paper, a new PSO architecture with a two-level pipeline is proposed to overcome this difficulty. The proposed structure aims to improve the flexibility of PSO hardware and further enhance the calculation performance. Therefore, two distinctive features are adopted in the proposed hardware, which are a generic particle calculation block (GPCB) [9] and a two-level pipeline approach. These two features will be introduced comprehensively in the following content.

The rest of this paper is organized as follows. In Sect. 2, the concept of the PSO algorithm and the conventional hardware will be introduced in brief. In Sect. 3, details of the proposed PSO hardware with the two-level pipeline will be introduced. In Sect. 4, the simulation results of comparisons between the proposed hardware and conventional hardware are provided. Finally, in Sect. 5, a conclusion is given.

2. Related Works

2.1 PSO algorithm

The original PSO [1] algorithm was inspired by the social behavior of bird flocks, fish schools, bee swarms, and animal herds. In a particular particle swarm, each particle represents a candidate solution for an optimization problem. The particles move around in the parameter space

Figure 1: Conventional hardware architecture

while obeying systematic rules to adjust their velocities and positions in response to the swarm's experience in locating quality solutions. Therefore, the dynamics (velocity and position) of the particles in the swarm are influenced by a combination of two factors: the personal and social best positions. The behavior process is defined by the following equations:

$$v_{id}^{t+1} = w * v_{id}^t + c_1 * r_1 * (p_{id} - x_{id}^t) + c_2 * r_2 * (p_{gd} - x_{id}^t) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (2)$$

where v_{id} and x_{id} are the velocity and position of a particle, respectively, p_{id} is the personal best position, and p_{gd} is the global best position. Both of these positions are selected depending on the calculation of fitness, which is defined specifically in different applications. c_1 and c_2 are acceleration constants, w is the inertia weight factor, and d is the number of dimensions in the search space.

Once the first iteration is completed, the personal and global best positions are updated. New values for the personal and global best positions result in variation of all the particles' velocities and positions during the next iteration. This process continues until all the iterations are finished and the final global best is considered to be the optimal solution of the system.

2.2 Conventional hardware

The conventional hardware architecture introduced by Cai et al. is shown in Fig. 1. A particle calculation block (PCB) is used to implement the type of PSO algorithm named PSO-RTVIWAC [7]. In order to achieve high-speed convergence, PSO-RTVIWAC involves a parameter automation strategy to perform complex calculations of the parameters. Combined with Eq. (1), the automation parameters are defined in the following equations:

$$w = r_3 * (w_{\max} - t * (w_{\max} - w_{\min}) / T) \quad (3)$$

$$c_1 = r_4 * (c_{\max} - t * (c_{\max} - c_{\min}) / T) \quad (4)$$

$$c_2 = r_5 * (c_{\max} - t * (c_{\max} - c_{\min}) / T) \quad (5)$$

$$x_{id} = k * v_{id}^{t+1} + x_{id}^t \quad (6) \quad phi = 4 * (1 + r_6) \quad (7)$$

$$k = 2 / \left| 2 - phi - \sqrt{phi^2 - 4 * phi} \right| \quad (8)$$

where t is the current generation and T is the total number of generations; k is a new velocity limitation factor introduced in Eq. (8).

In addition, on account of the low I/O operation speed of RAM, a pipeline approach is adopted in the conventional hardware to utilize the waiting time. Since the calculation of particles is executed iteratively, a particle calculation block can make use of the waiting time to process particle $n+1$, when fitness calculation block (FCB) is calculating the fitness of particle n . In the work of Cai et al., the calculation process is further divided into six phases: reading the input of the PCB, calculation of the PCB, storing the output of the PCB, reading the input of the FCB, calculation of the FCB, and storing the output of the FCB. To realize these operations, registers are assigned in each I/O port, as shown in Fig. 2. The above approach can increase the calculation speed.

Figure 2: Data path between registers

However, the conventional hardware only implements a particular PSO algorithm, which means that the hardware needs to be redesigned to meet different requirements in different PSO applications. In addition, the pipeline approach is only exploited in the reading and storing stages,

without sufficient consideration of the complex calculation of particle fitness values.

3. Proposed PSO Hardware

To overcome the above difficulties, a two-level pipeline structure for the PSO algorithm is proposed. The proposed hardware employs two distinctive features, a GPCB and a two-level pipeline approach, as shown in Fig. 3.

Figure 3: Proposed hardware architecture

The central part of the hardware is the control unit. It synchronizes the operation of each block and sends enabling signals to start the operation of each block. After receiving feedback signals, the control unit starts the operation of the next block. In addition, it can also monitor the operation of other blocks. The GPCB calculates the velocity and position of all particles, and sends the position information to the FCB to calculate the fitness value. RAM and ROM are used as storage blocks to store the calculation results and coefficients, respectively.

3.1 Generic particle calculation block

The GPCB is adopted to improve the flexibility of the proposed hardware, which means that the architecture of the hardware can be utilized in different applications without hardware modification. The GPCB is composed of several mathematical functions as follows:

$$v_{id}^{t+1} = sel_1 * [\omega * v_{id}^t + c_1 * r_1 * (p_{id} - x_{id}^t) + c_2 * r_2 * (p_{gd} - x_{id}^t) + sel_2 * c_3 * r_3 * (R_i^t - x_{id}^t)] \quad (9)$$

$$x_{id}^{t+1} = sel_3 * x_{id}^t + sel_4 * v_{id}^{t+1} + sel_5 * p_{id} + sel_6 * p_{gd} \pm sel_7 * \beta * |mbest - x_{id}^t| * \ln(1/r_4) \quad (10)$$

$$mbest = \sum_{i=1}^N \frac{p_{id}}{N} \quad (11)$$

where $sel_1 - sel_7$ are the particle motion coefficients,

which help to change the equations into different expressions for different PSO algorithms. In other words, the particle motion coefficients are adopted to combine the mathematical equations of each particular PSO algorithm into Eqs. (9)-(11).

3.2 Structure of two-level pipeline

The second feature is the two-level pipeline approach, which helps to divide the algorithm into several stages. The first-level pipeline is the same as that in the conventional hardware for data exchange operation. Owing to the large time consumption for accessing data from memory, the first-level pipeline is exploited to accelerate the I/O operation of RAM. Similarly to the pipeline in the conventional hardware, the calculation process is divided into six phases: reading the input of the GPCB, calculation of the GPCB, storing the output of the GPCB, reading the input of the FCB, calculation of the FCB and storing the output of the FCB, as shown in Fig. 4.

Figure 4: First-level pipeline

The second-level pipeline is designed for the calculation blocks. Since the GPCB and FCB are used for numerous mathematical functions, there are many redundant clock cycles while waiting for the calculation results from these blocks. In particular, the FCB is much more complex than GPCB, and the FCB needs to wait for the calculation result from the GPCB. Therefore, the FCB is divided into three parts named F1 to F3 to balance its calculation speed with the speed of GPCB, as shown in Fig. 5. This method greatly reduces the number of redundant waiting clock cycles in the calculation.

Figure 5: Second-level pipeline

Table 1: Comparison of simulation results

	Particle number	Iteration number	Logic elements	Calculation time for one particle	Total calculation time
Conventional hardware	10	50	3972	76.98ns	38.49 μ s
Proposed hardware	10	50	4167	34.83ns	17.41 μ s
Improvement ratio	–	–	–	121%	

4. Simulation Results

The proposed architecture is directly implemented on a Stratix II EP2S60F1020C3 development board with Verilog Hardware Description Language in Quartus II 9.0, similarly to the conventional hardware. The experimental conditions are set to 10 particles with 50 iterations. The experimental results show that the number of logic elements in the proposed hardware is 4,167 which is 4.9% higher than the number in the conventional hardware. However, the total calculation time of the proposed hardware is 17.41 μ s, while the conventional architecture needs 38.49 μ s to find its target, which means that the speed is increased by about 121% compared with that of the conventional hardware, as shown in Table 1.

5. Conclusions

This paper proposed a serial hardware architecture for the particle swarm optimization (PSO) algorithm with a two-level pipeline approach based on field-programmable gate array (FPGA) technology. In contrast to the conventional method, the proposed architecture employs a generic particle calculation block (GPCB) to support various kinds of PSO algorithm, which means that it is suitable for different applications. Furthermore, a two-level pipeline structure is adopted to increase the calculation speed. The first level is for data storage and the second level is designed for the calculation unit. The experimental results confirm that the speed of the proposed hardware is increased by 121% compared with that of the conventional hardware.

Acknowledgment

This work was supported by a Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research (No. 26280017).

References

[1] J. Kennedy and R. C. Eberhart: Particle swarm optimization, Proc. IEEE Int. Conf. Neural Networks, pp. 1942-1948, Nov. 1995.

[2] Y. Shi and R. C. Eberhart: A modified particle swarm optimizer, Proc. IEEE Int. Conf. Evolutionary Computation, pp. 69-73, May 1998.

[3] D. Parrott and X. Li: Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Trans. Evol. Comput., Vol. 10, No. 4, pp. 440-458, 2006.

[4] A. Farmahini-Farahani, S. Vakili, S. M. Fakhraie, S. Safari and C. Lucas: Parallel scalable hardware implementation of asynchronous discrete particle swarm optimization, Int. J. Intell. Real-Time Autom., Vol. 23, No. 2, pp. 177-187, 2010.

[5] X. Cai, S. Ngah, H. Zhu, Y. Tanabe and T. Baba: Pipeline architecture of particle swarm optimization, Proc. 9th IEEE/ACIS Int. Conf. Computer and Information Science, pp. 3-8, Aug. 2010.

[6] D. M. Munoz, C. H. Llanos, L. dos S. Coelho and M. Ayala-Rincon: Hardware particle swarm optimization with passive congregation for embedded application, Proc. VII Southern Conf. Programmable Logic, pp. 173-178, Apr. 2011.

[7] H. Zhu, Y. Tanabe and T. Baba: A random time-varying particle swarm optimization for the real time location systems, IEEJ Trans. Electron. Inf. Syst., Vol. 128-C, No. 12, pp. 1747-1760, 2008.

[8] K.-T. Chen, S. Ngah, H. Zhu and T. Baba: Implement random time-varying particle swarm optimization for positioning system, 2009 RISP Int. Workshop Non-linear Circuits and Signal Processing (NCSP'09), pp. 9-12, Mar. 2009.

[9] K.-T. Chen, M. Jia, J. Zhang and T. Baba: A flexible hardware architecture for particle swarm optimization, J. Signal Process., Vol. 16, No. 6, pp. 519-526, 2012.