A Flexible Hardware/Software Codesign for Particle Swarm Optimization

Kui-Ting Chen, Xiao Wu, Molin Jia and Takaaki Baba

Graduate School 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: nore@akane.waseda.jp, littlesea10ban@ruri.waseda.jp, jiamolin717@akane.waseda.jp, tbaba@waseda.jp

Abstract

The conventional hardware/software codesign cannot meet the requirement for a PSO algorithm that needs to be changed for different applications. In addition, the architecture of the conventional hardware/software codesign leads to a low processing speed. This paper presents a flexible hardware/software codesign to support various PSO algorithms and increase the processing speed of the conventional hardware/software codesign for various PSO applications. By adopting a simplified hardware architecture, the calculation speed can be improved by reducing the communication overhead. Furthermore, an improved generic particle calculation block (GPCB) is used to enhance the flexibility. It selects an appropriate PSO algorithm without the need for hardware modification to further increase the processing speed. The experimental results proved that the proposed flexible hardware/software codesign can achieve high processing speed and high flexibility with low chip cost.

1. Introduction

Particle swarm optimization (PSO) is a powerful algorithm for finding an optimal solution in a complex search space. The PSO algorithms have been widely used in many applications. The main advantages of the PSO algorithm are that it can produce excellent results with a reasonable resource cost and it is easy to implement in software. Some researchers proposed various PSO algorithms and proved them to be satisfactory in real applications. However, the PSO algorithm requires a long time even when it is executed on a high-performance workstation [1]. PSO hardware implementation is a significant approach for reducing the execution time of its software implementation.

Some PSO hardware architectures have been proposed [1-3]. In our previous research [1, 3], a pure PSO hardware with an extremely high processing speed was proposed. However, it can only be used for one particular fitness calculation. Pure hardware of the PSO algorithms with high flexibility, which can automatically redesign its hardware architecture for different applications, is expected in the near future. In order to achieve this goal, flexible hardware/software codesign is important work for realizing flexible pure hardware.

A conventional hardware/software codesign [2] of the PSO algorithm was proposed for solving the above problem. The fitness calculation is established in an on-chip processor with software, and other calculations are implemented in the hardware. Compared with the pure hardware, the conventional codesign can easily redesign the fitness calculation for different applications. The disadvantage of the conventional codesign is that the hardware calculation can only be used for one particular PSO algorithm, as shown in Fig. 1(a). In most cases of PSO applications, the type of PSO algorithm must be changed to achieve higher processing speed. A flexible hardware/software codesign is proposed for supporting various PSO algorithms to achieve higher processing speed in different applications, as shown in Fig. 1(b).

(a) Drawback of conventional hardware/software codesign

(b) Advantage of flexible hardware/software codesign

Figure 1: Comparison of two hardware/software codesigns

The rest of this paper is organized as follows. In Section 2, the original PSO concept and the conventional hardware/software codesign are briefly introduced. In Section 3, the details of the proposed hardware/software codesign are presented. In Section 4, the experimental results of the proposed and conventional approaches are provided. Finally, Section 5 comprises a summary and the
conclusions of this research.

2. Related Works

2.1 Particle swarm optimization

PSO is a relatively new sociologically inspired stochastic optimization algorithm based on swarm intelligence and was introduced by Kennedy and Eberhart in 1995 [4]. The basic operation of the PSO algorithm is updating the position and velocity of particles to find an optimal solution. Each particle \( i \) has a current velocity \( v_i \) and a personal best position \( p_{id} \) that represents a possible solution of optimization space. Considering an \( N \)-dimensional evaluation function, the position and velocity of particle \( i \) in the \((t+1)\)th iteration are updated by the following equations:

\[
\begin{align*}
    v_{id}' &= \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \\
    x_{id}' &= x_{id} + v_{id}'
\end{align*}
\]

where \( r_1 \) and \( r_2 \) are uniformly random numbers in the range \([0,1]\), \( p_{gd} \) is the location of the particle when the best fitness value is obtained for the whole population, \( c_1 \) and \( c_2 \) are two acceleration constants, \( \omega \) is called the inertia weight factor, and \( d \) is the number of dimensions in the problem space.

2.2 Conventional hardware/software codesign for PSO

The conventional hardware/software codesign [2] of the PSO algorithm was proposed. This codesign consists of an on-chip processor and particle updating accelerator, as shown in Fig. 2. In order to reduce the calculation time of software, particle re-initialization and fitness comparison are implemented in the particle updating accelerator. However, this approach leads not only to increase chip cost but also to an extra communication cost of the on-chip processor. The communication cost must be considered in the conventional hardware/software codesign. Furthermore, the particle updating accelerator only supports a particular PSO algorithm. It is difficult to meet the different requirements for PSO applications.

3. Proposed Flexible Hardware/Software Codesign for PSO

The proposed flexible hardware/software codesign exploits two features to improve the hardware flexibility and processing speed, as shown in Fig. 3. One is simplified hardware architecture for increasing the processing speed of the conventional hardware/software codesign. The other feature is the use of an improved generic particle calculation block (GPCB) for supporting the various PSO algorithms to meet the requirements of PSO applications.

3.1 Simplified hardware architecture

In the conventional hardware/software codesign [2], the particle updating accelerator adopts a parallel computation approach. Such approach requires a long communication time to exchange data with the on-chip processor, since a fitness comparison module and particle re-initialization module increase the burden of the data bus. To compare the fitness value of each particle, the related information of the particles must be transmitted to the fitness comparison module. This approach requires more time than comparing the fitness value using an on-chip processor. In order to reduce communication cost, some hardware modules of the particle updating accelerator must be implemented into the on-chip processor. However, this slightly increases the processing time of software as a trade-off.

To balance the communication time and processing time of software, a simplified hardware/software codesign is proposed. In the proposed approach, the fitness comparison module and the particle re-initialization module are implemented by software to balance the communication time and processing time, as shown in Fig. 3. This approach can reduce the amount of communication time needed to exchange the related information of each particle. As a trade-off, the fitness calculation block (FCB) is needed to compare the fitness values of the particles for evaluating \( p_{gd} \) and \( p_{id} \). Furthermore, the parallel calculation approach of the hardware accelerator module is changed to the serial calculation approach to reduce the
chip cost of the conventional hardware/software codesign.

3.2 Improved generic particle calculation block

In Fig. 2, the PSO core is used to calculate the position and velocity of each particle by implementing Eqs. (1) and (2). Therefore, the PSO core can only support one particular PSO algorithm. However, most PSO applications required different PSO algorithms for achieving high performance [3]. A generic particle calculation block (GPCB) is proposed to support at least 10 PSO algorithms [3] (original PSO, quantum-behaved PSO (QPSO), PSO with a random time-varying inertia weight factor and acceleration coefficients (PSO-RTVIWAC), and so on). By adopting the concept of the GPCB into the proposed hardware/software codesign, it not only can support various PSO algorithms but also can achieve higher processing speed by selecting an appropriate PSO algorithm for different PSO applications. The improved GPCB adopts a 32-bit fixed-point numerical system for reducing the communication time and chip cost.

4. Experimental Results

The proposed hardware/software codesign has been developed in Quartus II 7.2 with Altera Cyclone II FPGA EP2C70F896, using the verilog hardware description language. For comparison, all the parameters are set at the same values as in [2], as shown in Table 1. In addition, the same conditions of four benchmark fitness functions are used in the experiments to evaluate the proposed hardware/software codesign, as shown in Table 2.

4.1 Flexibility and processing speed

In the comparison of calculation time, the four benchmark fitness functions were used to evaluate the processing speed for two different hardware/software codesigns under three different conditions of particle numbers (P), as shown in Table 3. The experimental results in Table 3 were produced using the original PSO algorithm. The experiments were terminated once deviation between the global best p_gd and the optimal value became less than $10^{-4}$. The optimal values of each fitness function are shown in Table 2. In the first experiment, the particle number and maximal number of iterations were set at 8 and 1,250. The processing times of the proposed hardware/software codesign were 0.0391, 0.1627, 0.0483, and 0.0142 seconds for evaluating $f_1(x)$, $f_2(x)$, $f_3(x)$, and $f_4(x)$, respectively. In the second experiment, the processing times increased from 16 to 32, almost the same speed improvement.

### Table 1: Experimental parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Particle (P)</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Termination Error Threshold</td>
<td>$&lt;10^{-4}$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>0–2</td>
</tr>
<tr>
<td>$c_2$</td>
<td>0–2</td>
</tr>
<tr>
<td>$\omega$</td>
<td>0.25</td>
</tr>
<tr>
<td>System Clock</td>
<td>50 MHz</td>
</tr>
<tr>
<td>Maximal Number of Iterations</td>
<td>1250</td>
</tr>
</tbody>
</table>

### Table 2: Benchmark fitness functions

<table>
<thead>
<tr>
<th>Fitness Functions</th>
<th>Size of Search Space</th>
<th>Optimal Value</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1(x) = \left[1+\frac{(x_1 + x_2 + 1)^2 + (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) + (30 + (2x_1 - 3x_2)^2 + (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))}{100}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table><p>ight] - 2 &lt; x_1, x_2, i = 1,2,3,4 | $-2 &lt; x_1, x_2, i = 1,2,3,4$ | 3 | 2 |
| $f_2(x) = \frac{\sum_{i=1}^{n}(x_i - 1)^{2} + \sum_{i=1}^{n}(x_i - 1)^{2}}{100} + \frac{\sum_{i=1}^{n}(x_i - 1)^{2}}{100}$ | $-9 &lt; x_i, i = 1,2,3,4$ | 0 | 4 |
| $f_3(x) = 100(x_1 - x_2)^2 + (1 - x_2)^2$ | $-9 &lt; x_i, i = 1,2$ | 0 | 2 |
| $f_4(x) = x_1^2 + x_2^2 + x_3^2$ | $-5.12 &lt; x_i &lt; 5.12, i = 1,2,3$ | 0 | 2 |</p>

### Table 3: Processing speeds

<table>
<thead>
<tr>
<th>$P$</th>
<th>Hardware Approach</th>
<th>$f_1(x)$</th>
<th>$f_2(x)$</th>
<th>$f_3(x)$</th>
<th>$f_4(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Proposed</td>
<td>0.0391</td>
<td>0.1627</td>
<td>0.0483</td>
<td>0.0142</td>
</tr>
<tr>
<td>16</td>
<td>Conventional</td>
<td>0.1468</td>
<td>2.0773</td>
<td>0.2283</td>
<td>0.0701</td>
</tr>
<tr>
<td></td>
<td>Speed up (Times)</td>
<td>3.7</td>
<td>12.7</td>
<td>4.7</td>
<td>4.9</td>
</tr>
<tr>
<td>32</td>
<td>Proposed</td>
<td>0.0484</td>
<td>0.2643</td>
<td>0.0718</td>
<td>0.0203</td>
</tr>
<tr>
<td></td>
<td>Conventional</td>
<td>0.1747</td>
<td>3.2955</td>
<td>0.3162</td>
<td>0.0946</td>
</tr>
<tr>
<td></td>
<td>Speed up (Times)</td>
<td>3.6</td>
<td>12.4</td>
<td>4.4</td>
<td>4.6</td>
</tr>
</tbody>
</table>

- $f_1(x)$: Rosenbrock function
- $f_2(x)$: Rastrigin function
- $f_3(x)$: Griewank function
- $f_4(x)$: Schwefel function
ments were confirmed. Table 3 proves that the processing speed of the proposed hardware/software codesign can be 3.6 times to 12.8 times higher than that of the conventional approach. In particular, with increasing dimension of the fitness function, the performance of the proposed hardware/software codesign is further improved.

In the second experiment, an appropriate PSO algorithm was selected by the improved GPCB for different fitness functions. The particle number in this experiment was set at 16, since most PSO applications require a particle number between 10 and 20. The processing speeds of \( f_1(x) \), \( f_2(x) \), and \( f_3(x) \) were further enhanced to 34.3, 9.3, and 7.1 times that of the conventional approach by selecting QPSO and PSO-RTVIWAC, as shown in Table 4. (The processing speed of \( f_4(x) \) was not enhanced since the original PSO is the most suitable algorithm.) Therefore, selecting an appropriate PSO algorithm can enhance the performance of different fitness functions. The experimental results prove that the proposed flexible hardware/software codesign achieves high flexibility and high processing speed simultaneously.

### 4.2 Chip cost

A Nios II soft-core processor [5] was implemented as the on-chip processor in both designs. The chip cost and reduction ratio between the proposed and conventional approaches are presented in Table 5. The proposed hardware/software codesign reduced the chip cost by 23.4%, 50.2%, and 68.4%. By increasing the particle number, the chip cost of the conventional hardware/software codesign increased from 10,063 logic elements (LEs) to 24,383 LEs. In the case of the proposed flexible hardware/software codesign, the chip cost was fixed to 7,710 LEs. Compared with the conventional codesign, the proposed codesign can achieve higher processing speed and higher flexibility with lower chip cost.

### 5. Conclusions

In this paper, a flexible hardware/software codesign is proposed to improve the flexibility and increase the processing speed of the conventional codesign. The proposed codesign employs two features. One is a simplified hardware architecture that can significantly reduce the communication time between hardware and software to increase the processing speed of the proposed approach. The other one is an improved generic particle calculation block (GPCB) that can improve the flexibility of the hardware modules. It can select an appropriate PSO algorithm to further increase the processing speed. The experimental results proved that the simplified hardware architecture of the proposed hardware/software codesign can achieve about 3.6 to 12.8 times higher processing speed than that of the conventional codesign. The processing speed can be further improved to 34.3 times higher than that of the conventional codesign by using improved GPCB that can select an appropriate PSO algorithm for a particular fitness function. In addition, the proposed flexible hardware/software codesign can reduce the chip cost by 23.4% to 68.4%. The proposed codesign will be a favored solution to achieve higher flexibility and higher processing speed with lower chip cost.

### References


