Send Orders for Reprints to reprints@benthamscience.ae

The Open Cybernetics & Systemics Journal, 2015, *9,* 1145-1154

Open Access

Research on an Improved Algorithm for 3D NoC Floorplanning Based on Particle Swarm Optimization

Zhang Dakun[1,*], Song Guozhi[1], Wang Lianlian[2] and Huang Cui[1]

[1]*School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin300387, China;* [2]*Network Center, Tianjin Agricultural University, Tianjin 300384, China*

Abstract: Nowadays, three-dimensional network-on-chip (3D NoC) with its shorter global interconnects, higher performance, lower loss of interconnection, higher packing density, smaller volume, and many other advantages, has drawn more and more attention in both industrial and academic circle. In this paper, an improved algorithm, named the algorithm based on particle swarm optimization algorithm to optimize the floorplans (PSO-NoC), has been proposed with simulations conducted to rectify this algorithm. The simulation results are compared with the original Simulated Annealing-NoC. The experimental results show that the PSO-NoC algorithm reduces the latency and improves the throughput compared with the original one. Particularly, the CPU's process time is significantly decreased by 52.78% in the average case and 87.08% in the best case respectively.

Keywords: 3D NoC, floorplanning, improved algorithm, PSO.

1. INTRODUCTION

With the development of chip manufacturing and the size expansion of chips, the former bus architecture and point-to-point interconnection cannot satisfy the demand of on-chip communications any more. Therefore, the network-on-chip (NOC) has drawn much more attention because of its good scalability and parallel communication capability. However, due to the increased functions of chip, the number of integrated transistors multiplied two-dimensional network on chip has reached a bottleneck in the aspects of area, power consumption, layout, and packing density and so on. In this case, the 3D NoC came into being, with its shorter global interconnections, higher performance, lower interconnect losses, higher packing density, smaller size and many other advantages, has caused more and more attention in both industry and academia. Among many technic issues faced by 3D NoC researchers floorplanning is a crucial to the performance of the design process for 3D NoCs.

A floorplan of an IC is a schematic representation of tentative placement of major functional IP blocks in electronic design automation (EDA). Floorplans can be categorized into two groups, the sliceable floorplans [1-3] and non-sliceable floorplans [4-7]. For examples, a team from Tsinghua University led by She-Qin Dong mainly focused on the buffer insertion algorithm for interconnect centric floorplanning [8] and 3-D floorplan representation based on the methodologies of Corner Block List (CBL) [9]. Their contribution also included a divide-and-conquer 2.5-D floorplanning algorithm [10]. Liang-Li He *et al.* in [11] conducted a thorough study of virtual layout and proposed a FOB based man-machine interactive loading layout method. Another team led by Young, Evangeline F.Y. from Hong Kong university of science and technology delved into the problem of bus-driven floorplanning [12] and 3-D floorplanning using labeled tree and dual sequences [13]. Jingjing in [14] proposed a space feature optimization based mapping layout algorithm and they make a great contribution on energy consumption optimization of three dimensional networks-on-chip. Kakoee Mohammad Reza and Angiolin Federico *et al.* in [15] proposed a floorplan-aware toolchain for NoC design and synthesis integrated with a graphical front-end. They showed that not only a great amount of time and effort can be saved thanks to the easy-to-use proposed environment, but also that the quality of the final netlist can be improved due to the optimizations unlocked by the early-stage interaction among the designer and the proposed toolchain. They presented a floorplan-aware toolchain for NoC design and synthesis integrated with a graphical front-end, The resulting design methodology is highly automated yet entails rich interaction with the user, spanning across traffic flow specification, topology synthesis and physical floorplanning, with back-annotation capabilities and opportunities for incremental design. de Paulo, V and Ababei, C worked on homogeneous networks over heterogeneous floorplans [16]. They proposed a design methodology consisting of floorplanning and router assignment in a specifically designed tool that integrates a cycle accurate NoC simulator. It is implemented and then used to investigate the new architecture and show that experimental results are application specific with potential significant performance improvements for some testcases.

In 1995, Dr. James Kennedy and Russell Eberhart proposed Particle Swarm Optimization (PSO) algorithm [17, 18], and then PSO is widely used in various fields of social life and scientific research. Hao Pan in Wuhan University of

1874-110X/15 2015 Bentham Open

Technology applied PSO to the neural network training [19]. The number of dimensions of each particle is seen as system parameters of neural network studied. Run the PSO and you'll find the global optimal solution, and then you can find the corresponding optimal parameters optimal solution to achieve the purpose of training the neural network. In addition, Guoan Liu in Harbin Institute of Technology and Xuan Gao in Xi'an Electronic Science and Technology University applied PSO to image retrieval [20, 21]. Quan Lin in Changsha University of Science and Technology and Guangzhou Chen in Hefei University of Technology applied PSO to the field of projection pursuit [22, 23]. In this paper, we first apply the classical PSO to 3D NoC floorplanning algorithm and open a new field of PSO application.

2. ALGORITHM BASED ON PARTICLE SWARM OPTIMIZATION ALGORITHM TO OPTIMIZE THE FLOORPLANS

2.1 PSO-NoC Algorithm Design

In this paper, an improved algorithm named the algorithm based on particle swarm optimization algorithm to optimize the floorplans (PSO-NoC) has been proposed. The algorithm is an improvement of the original algorithm based on simulated annealing algorithm to optimize the floorplans (SA-NoC) [24], using the parallel computing features of PSO to mainly optimize CPU processing time and make it more suitable for floorplanning on 3D NoCs with a large-scale topology.

The cost function of PSO-NoC algorithm used is shown in Formula (1),

$$Costfunction = \alpha \cdot Area + (1-\alpha)WireLength \quad (1)$$

In (1), α represents the cost of completing the layout needs. $Area$ represents areas of IP cores layout using. $WireLength$ represents length of the connection of generating layout uses. α is a parameter specified by the user which used to balance the area of tiles and the length of layout uses and valued from 0 to 1. In our algorithm, the value of α is 0.25. The update formulas of velocity and position in our algorithm are shown as Formula (2) and Formula (3),

$$v_{i+1} = \omega v_i + c_1 \cdot d_1(pre\_cost_i - x_i) + c_2 \cdot d_2(best_i - x_i) \quad (2)$$

$$x_{i+1} = x_i + v_i \quad (3)$$

In Formula (2), inertia weight ω uses variable weight and calculated formula is shown as Formula (4). The initial value of ω_{min} and ω_{max} are 0.5 and 3. The parameter $count$ represents the current number of iterations and N represents the maximum number of iterations. Learning factor c_1 and c_2 use synchronous time-varying way and the calculated formula is shown as Formula (5). The initial value of c_{min} and c_{max} are 0.25 and 3. Random parameters d_1 and d_2 are numbers that program randomly generated from 0 to 1. The parameter pre_cost_i represents the best position of the particle itself and $best_i$ represents the optimum position of particle swarm, wherein the velocity is in the range [3, 3]. If the value exceeds the determined range, then revise it to equal to the maximum. The position sequence of particles satisfies with binary code. It also needs to be revised if it out of the determined range. If the location information obtained is less than 0.5, then record it as 0, else record it as 1.

$$\omega = \omega_{max} - \frac{(\omega_{max} - \omega_{min}) * count}{N} \quad (4)$$

$$c_1 = c_2 = c_{max} - \frac{(c_{max} - c_{min}) * count}{N} \quad (5)$$

2.2. PSO-NoC Algorithm Realization

In this part, we show the realization of PSO-NoC algorithm. The steps are as follows.

Step 1: Parameters initialization. Define the number of particles as shu and the maximum number of iterations as N, where $N = times * fp\_p \rightarrow size()$. Randomly generate shu initial particles X_0 and velocity V_0 which are from -3 to 3.

Step 2: Assume the best position of the particle itself is pre_cost and the global best position of particle swarm is $best$.

Step 3: Decide whether the current iteration number reaches the maximum number N. If it achieved, then jump to Step 6, else jump to step 4.

Step 4: Do the following for all particles:

⊖ Calculate the fitness of each particle according to Formula (1). If the particle adapt better than pre_cost, then the value is assigned to pre_cost and update the best position of the individual particle.

⊜ If the fitness of pre_cost is better than $best$, then update the value of $best$ and the value of $goodnum$ adds to 1, else the value of $badnum$ adds to 1.

⊛ Update the position and velocity of particles according to Formula (2) and Formula (3).

④ Correct the obtained position information of particles according to the determined range of value of position and velocity so that it would not exceed the available space.

Step 5: The number of iterations minus 1 and jump to Step 3.

Step 6: Output *best* and end the algorithm.

2.3. Flowchart of PSO-NoC Algorithm

The flowchart of PSO-NoC algorithm is shown as Fig. (**1**).

3. SIMULATION RESULTS AND DISCUSSION

3.1. Introduction of the Simulator

In order to test the degree of improvement of the proposed algorithm on CPU processing time, average flit la-

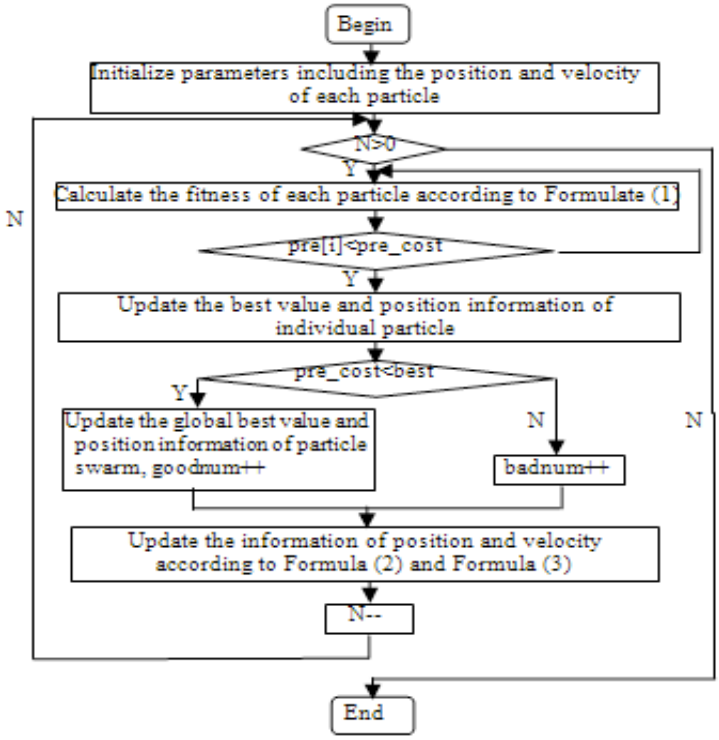

Fig. (1). Flowchart of PSO-NoC algorithm.

Table 1. Characteristics of the used testcases.

Testcases	Number of IP/cores	Core avg. W/H	Core std. dev. of W/H	Direct Topology R×R
apte	8	4324/2499	27/4	3×3
xerox	10	2114/2872	335/1290	4×4
hp	11	4533/924	2498/386	4×4
ami25	25	1770/1408	1201/896	5×5
ami33	33	1581/1573	830/865	6×6
ami49	49	1089/1123	768/651	7×7

tency and network throughput, the algorithm needs to be run on an actual system and compared with the original algorithm.

In this paper, we use the VNOC3 simulation platform developed by Cristinel Ababei in North Dakota State University using C++ in the Linux system. The simulator is actually a platform used to study 3D NoC architecture with two or three layers. The platform is built on a previous version of VNOC (a simulator for 2D NoC) and a B* tree layout. For three-layer architecture, the framework uses hMetis division. In addition, the simulator also includes a hidden option which allows users to generate new test cases and a GUI drawing tools. The drawing tools can be used to generate a three-layer network architecture but with only 2D display.

3.2. Testcases

In our experiments, we used six testcases whose characteristics are shown in Table **1**. In this table, we also present the size of the direct topologies for the testcases are all 3d NoC with heterogeneous network architecture. We constructed these testcases from the classic MCNC testcases, whose area was scaled to achieve an average size of about 1cm×1 cm, which is a typical area for NoCs reported in the literature [25].

Based on these six testcases, a lot of experiments have been conducted and the average of experimental results was calculated. But due to the limitation of paper space, we show only three testcases of the six to illustrate trends and the improvement of the performance. They are *apte*, *hp* and *ami49*.

We choose *apte* and *ami49* here for they respectively are the smallest and largest testcases in these six testcases and are more persuasive. The testcase *hp* is special, because the number of its IP core is relatively small (only 11). But its aspect ratio is 2 to 3 times the other testcases. So in this paper, we show the comparison in the next session.

3.3. Simulation Results and Analysis

With the simulator introduced above we implement PSO-NoC and compare it with the original SA-NoC. In the simu-

lation data, the performance of the algorithm is mainly measured by three parameters, *i.e.* CPU processor time, throughput and average flit latency. In the simulation experiments, in order to ensure a fair comparison, the value of command line parameters (for example, operating cycle, preheat cycle, input and output buffer size, *etc*) of the two algorithm is always consistent. In certain load conditions or of a certain Mesh size, the lower the average latency, higher the throughput, less the CPU processing time, the better performance of the algorithm.

In order to comprehensively study the impact of each indicator of 3D NoC on the performance, in the simulation experiment, we compare the throughout, average latency and CPU processing time from four aspects while other parameter values remain unchanged. (1) Change the number of virtual channels (from 2 to 6) and analyze the impact of virtual channels on average flit latency, throughput and CPU processing time. (2) Change the buffer size (from 1x to 5x and the data indicates 1 to 5 times the default cache) and analyze the impact of buffer size on average flit latency and throughput. (3) Change the injection load (20%-100%) and analyze the changes of average flit latency, throughput and CPU processing time. (4) Change Mesh size of the architecture (number of IP cores and aspect ratio) and analyze their impact on average flit latency, throughput and CPU processing time. Detailed comparison and analysis are seen in section A, B, C, D. In the experiment, a number of simulations have been conducted in each case and the final result is calculated from average. In each simulation, total run cycle is set to 60000cycles and preheat cycle is set to 1000cycles to ensure the obtained data values when the system reaches a steady state to reduce the error data obtained in the simulation.

3.3.1. The Impact of the Number of Virtual Channels on Performance

First, we observe the impact of increasing the number on virtual channels upon average flit latency, throughput and CPU processing time. In this scenario, we observe changes of average flit latency, throughput and CPU processing time while changing the number of virtual channels (VC) from 2 to 6 and compare the changes when using PSO-NoC and SA-NoC.

For the testcases *apte*, *hp* and *ami49*, the comparison of average flit latency, throughput and CPU processing time of PSO-NoC and SA-NoC are shown in Figs. (**2-4**). From the three figures, we can observe that for the testcase *apte*, compared with SA-NoC algorithm, although the average flit latency of PSO-NoC algorithm increases by 1.14% in average, network throughput improves by 1.99% in average. The most obvious is that the CPU processing time decreases by 52.35% in average which saves energy greatly.

For the testcase *hp*, with the three figures, we can conclude that compared with SA-NoC algorithm, the average flit latency of PSO-NoC algorithm decreases by 3.67% in average and the network throughput improves by 8.07% in average. The most obvious is that the CPU processing time decreases by 84.14% in average which saves energy greatly.

For the testcase *ami49*, we can observe that compared with SA-NoC algorithm, the average flit latency of PSO-NoC algorithm decreases by 12.99% in average, network throughput improves by 0.82% in average. The most obvious is that the CPU processing time decreases by 47.64% in average. So for the testcases *hp* and *ami49*, every performance of PSO-NoC algorithm is better than the original SA-NoC algorithm.

In summary, it can be seen that compared with the original SA-NoC, although latency of PSO-NoC is not better in some cases, average flit latency reduces 2.8% and throughput increases 2.25% as a whole. Especially, CPU processing time decreases greatly, by an average of 47.71% and the maximum case decreases 86.52%.

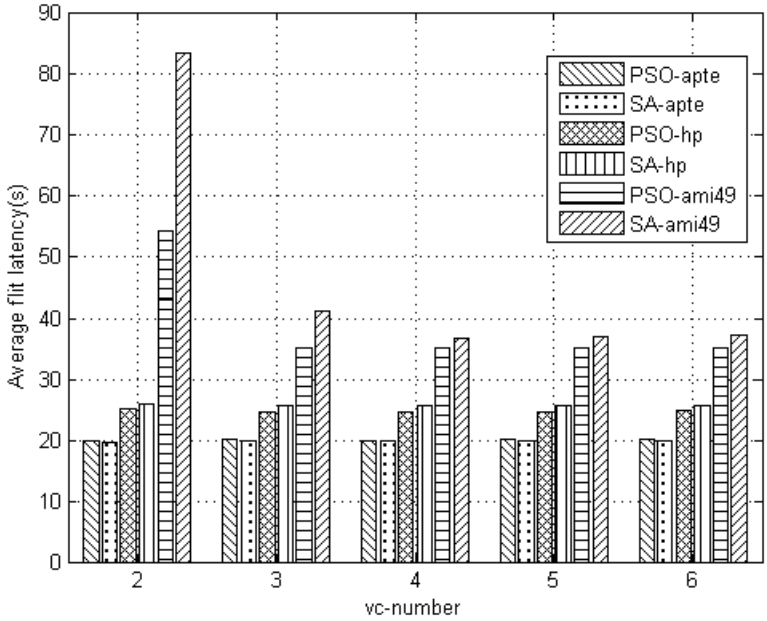

Fig. (2). Comparison of average flit latency for *apte*, *hp* and *ami49* while changing the number of virtual channels from 2 to 6.

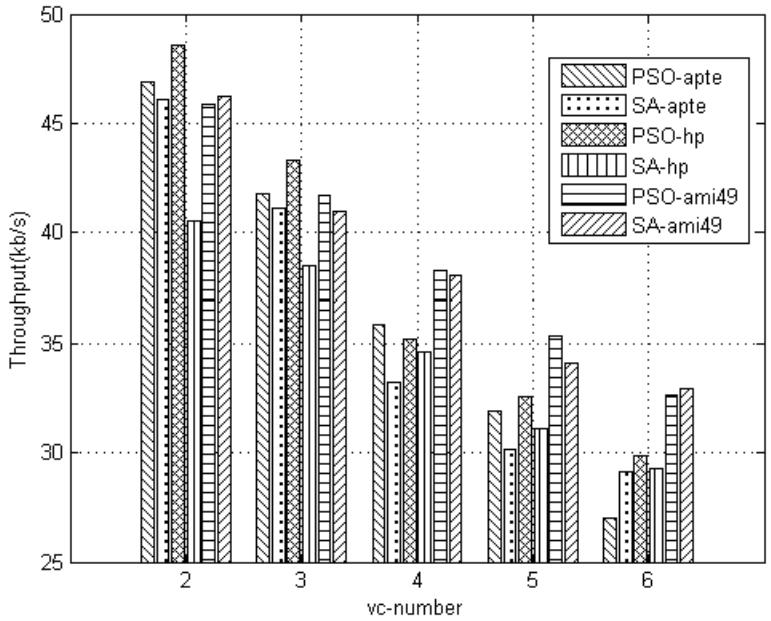

Fig. (3). Comparison of throughput for *apte*, *hp* and *ami49* while changing the number of virtual channels from 2 to 6.

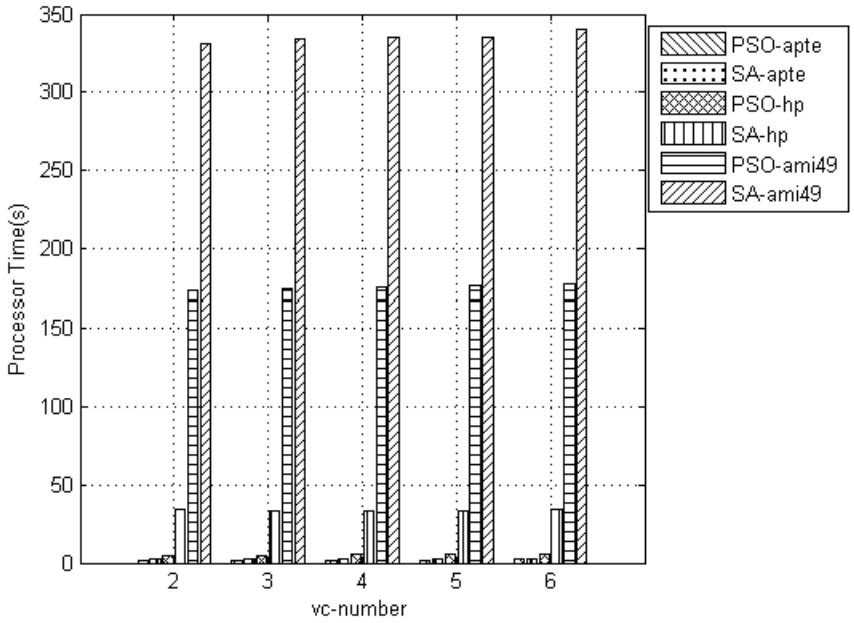

Fig. (4). Comparison of CPU processing time for *apte*, *hp* and *ami49* while changing the number of virtual channels from 2 to 6.

3.3.2. The Impact of Buffer Size on Performance

In this scenario, instead of increasing the number of virtual channels, we study the impact of buffer size on performance by increasing the size of buffers. We assume that the areas of routers is about 20% of the total area, we can increase the area of each router to reach to 5 times of the original value by increasing the size of input buffer and output buffer in each router port. We can observe the impact of buffer size on average latency and throughput by changing the size of buffers.

For the testcases *apte*, *hp* and *ami49*, the comparison of average flit latency and throughput of PSO-NoC and SA-NoC are shown in Figs. (**5**) and (**6**). From the two figures, we can observe that for the testcase *apte*, compared with SA-NoC algorithm, although the average flit latency of PSO-NoC algorithm increases by 1.04% in average, network throughput improves by 2.29% in average.

For the testcase *hp*, with the two figures, we can conclude that compared with SA-NoC algorithm, the average flit latency of PSO-NoC algorithm decreases by 3.92% in average and the network throughput improves by 4.58% in average. In this case, every performance of PSO-NoC algorithm is better than the original SA-NoC algorithm.

For the testcase *ami49*, we can observe that although the network throughput decreased by 1.36% in average, the average flit latency decreases by 6.27% in average.

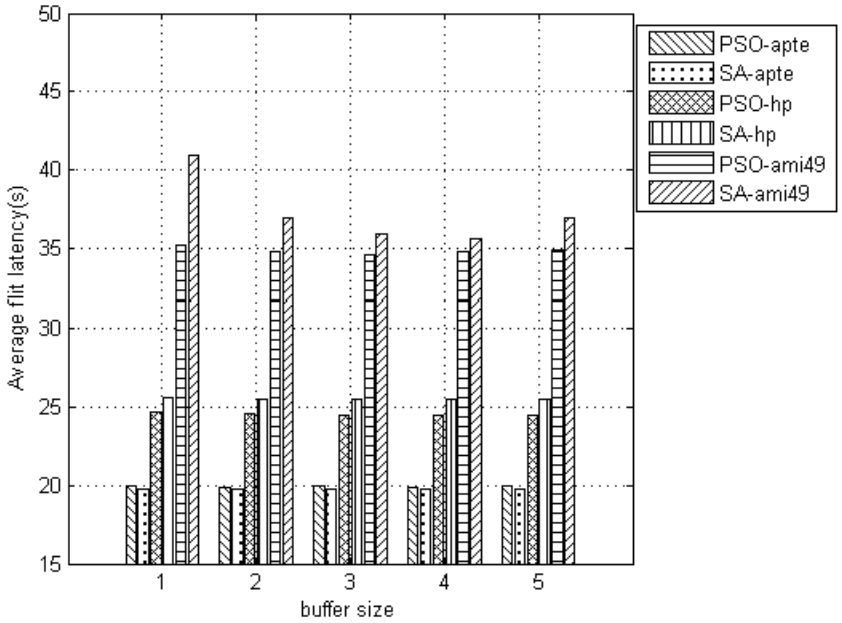

Fig. (5). Comparison of average flit latency for *apte*, *hp* and *ami49* while changing the size of buffers from 1x to 5x.

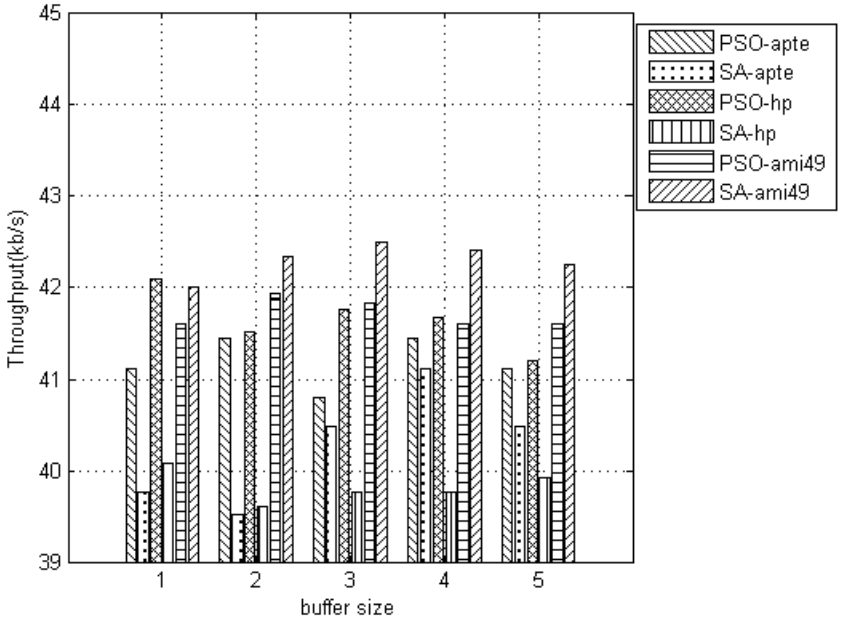

Fig. (6). Comparison of throughput for *apte*, *hp* and *ami49* while changing the size of buffers from 1x to 5x.

In summary, it can be observed that compared with the original SA-NoC, although latency of PSO-NoC are not better in some cases, average flit latency decreased by 7.51% and throughput increases by 5.3% as a whole.

3.3.3. The Impact of Injection Load on Performance

In fact, through a lot of experiments, we can conclude that the impact of changing the injection load on network latency and throughput is the largest.

For the testcases *apte*, *hp* and *ami49*, the comparison of average flit latency, throughput and CPU processing time of PSO-NoC and SA-NoC are shown in Figs. (**7-9**). From the three figures, we can observe that for the testcase *apte*, although the average flit latency increases by 0.86% in average, the average flit latency are substantially the same when the injection load reaches saturation and network throughput improves by 5.11% in average. The most obvious is that the CPU processing time decreases by 37.65% in average which saves energy greatly.

For *hp*, before injection load reaches 70%, average flit latency has no change substantially. Based on the three figures, we can conclude that although the average flit latency increases by 11.26% in average, the network throughput improves by 4.91% in average. The most obvious is that the CPU processing time decreases by 85.13% in average which saves energy greatly.

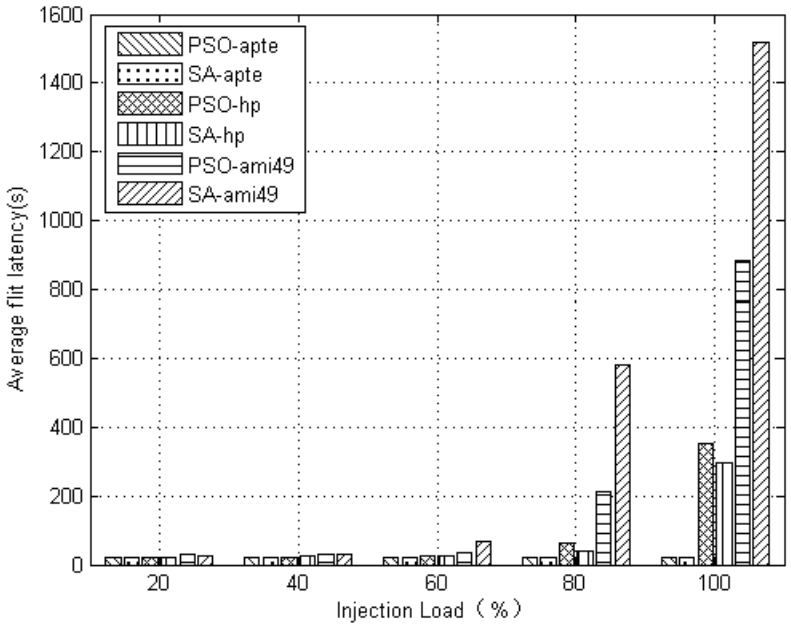

Fig. (7). Comparison of average flit latency for *apte*, *hp* and *ami49* while changing injection load.

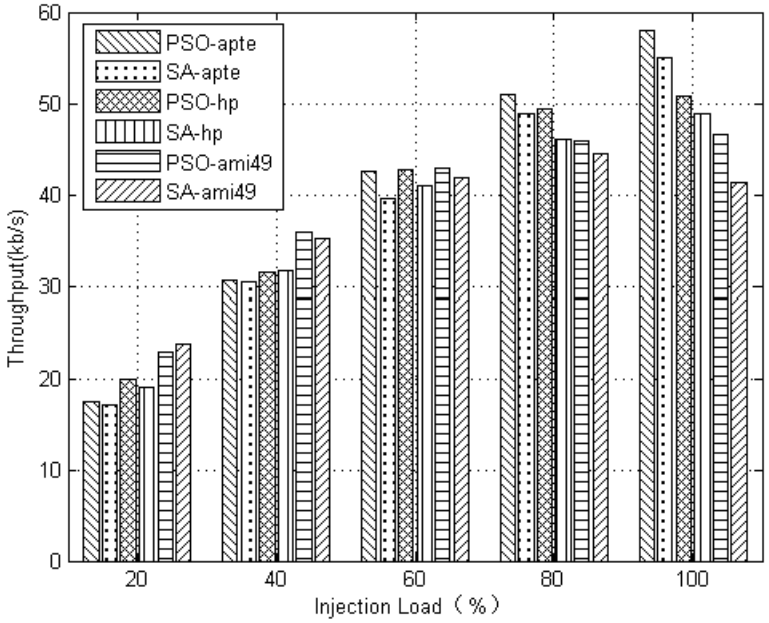

Fig. (8). Comparison of throughput for *apte*, *hp* and *ami49* while changing injection load.

For the testcase*ami49*, with the three figures, we can observe that compared with SA-NoC algorithm, the average flit latency of PSO-NoC algorithm increases by 27.72% in average, network throughput improves by 3.13% in average. The most obvious result is that the CPU processing time decreases by 49.28% in average. In this case, every performance of PSO-NoC algorithm is better than the original SA-NoC algorithm.

In summary, it can be seen that compared with the original SA-NoC, although latency of PSO-NoC is not better in some cases, average flit latency reduces 5.92% and throughput increases 4.41% as a whole. Especially, CPU processing time decreases greatly, by an average of 52.78% and the maximum case decreases 87.08%.

3.3.4. The Impact of Different Testcases on Performance

Different testcases have different numbers of IP cores and different aspect ratio of the IP cores. Also, the Mesh size of their direct topologies is different. In this section, we study the change of average flit latency, throughput and CPU processing time of different testcases. In order to get more equitable results, in this paper, simulations with different injection loads have been run and the results are collected as follows. When injection loads are different, the average flit latency increases as the number of IP cores increases, and the throughput is controlled by two parameters of Mesh size and aspect ratio. The variation trend of CPU processing time is substantially the same.

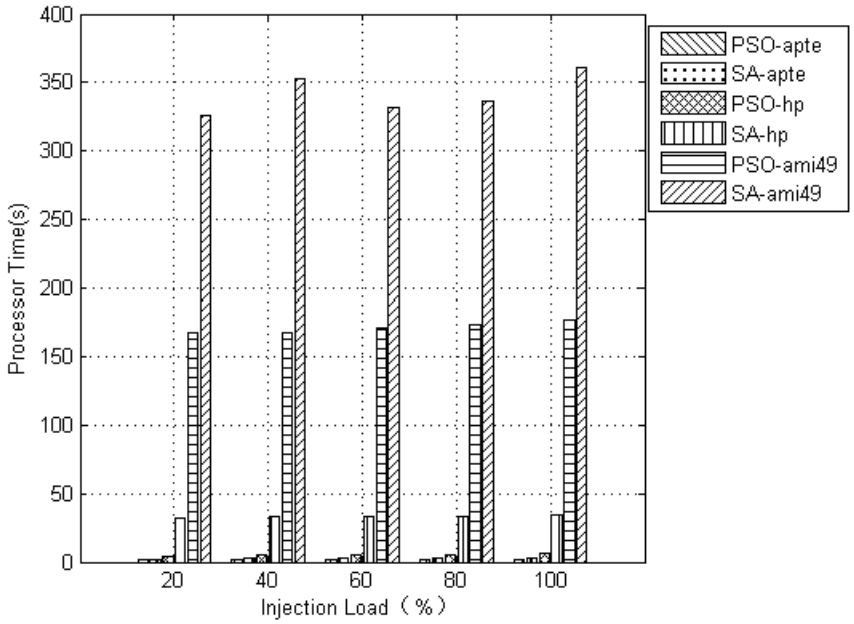

Fig. (9). Comparison of CPU processing time for *apte*, *hp* and *ami49* while changing injection load.

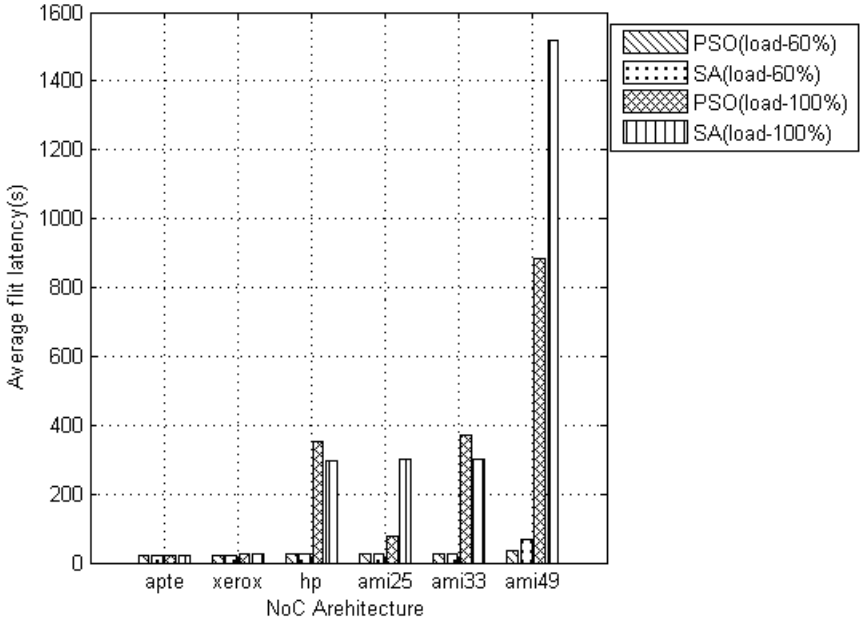

Fig. (10). Comparison of average flit latency of architectures with different Mesh size when injection load is 60% and 100%.

In this paper, we conduct simulation experiments of injection load of 20% to 100%, and then we get the bar chart and analyze it. But due to the limitations of our paper space, we just show the comparison of average flit latency in the condition of 60% and saturation injection load as shown in Fig. (**10**). It can be observed in the figures that when the Mesh size is less than 6, average flit latency increases more slowly and the average flit latency of the proposed PSO-NoC algorithm is mostly less than SA-NoC algorithm. When Mesh size is over 6, average flit latency of SA-NoC algorithm increases greater while that of the PSO-NoC algorithm increases slowly. Thus PSO-NoC algorithm is more suitable for network architecture with large Mesh size.

To illustrate the impact of different testcases on network throughput, in this paper, we also show the comparison of the impact of different Mesh size and respect ratio on throughput in the condition of 60% and saturation injection load as shown in Fig. (**11**). It can be observed in the figures that as a whole the throughput does not just increase or decrease as the number of IP cores increase, but shows a growth trend like a wavy line. This also explains why the throughput is not only affected by the number of IP cores but also the aspect ratio. According to the configuration parameters of each testcase, we can draw a conclusion that the throughput is proportional to the number of IP cores and the aspect ratio. Therefore, although the number of IP cores of

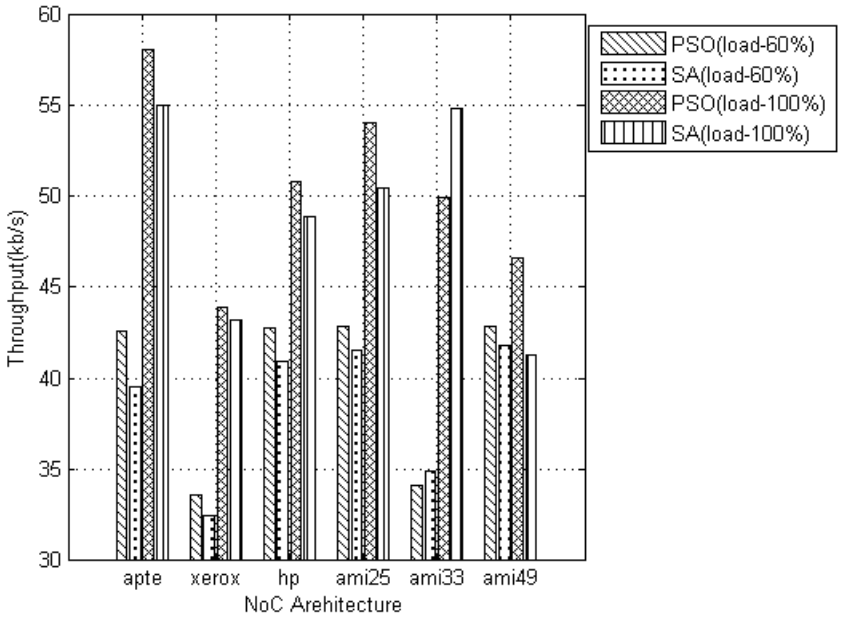

Fig. (11) Comparison of throughput of architectures with different Mesh size when injection load is 60% and 100%.

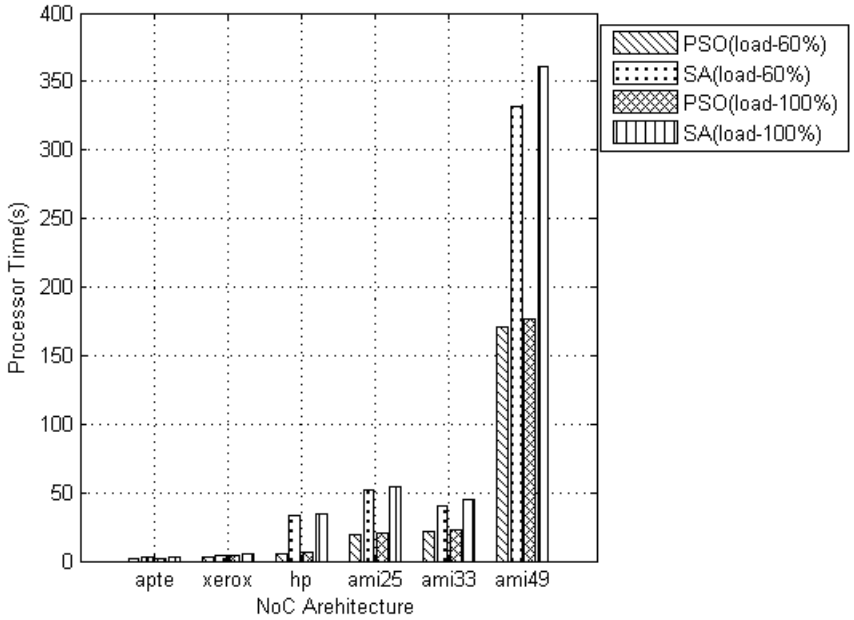

Fig. (12). Comparison of CPU processing time of architectures with different Mesh size when injection load is 60% and 100%.

xerox increases, the aspect ratio decreases close to 1. So the throughout presents a certain downward trend. For comparison of the two algorithms, the throughput of PSO-NoC algorithm is slightly higher than that of SA-NoC algorithm as a whole. In effect overall, PSO-NoC algorithm is better.

The figure of CPU processing time of each injection load is shown in Fig. (**12**). It can be observed in the figure that when the Mesh size is less than 6, CPU processing time increases more slowly. When Mesh size is over 6, CPU processing time of SA-NoC algorithm increases at a speed of 7 times while that of PSO-NoC algorithm increasing at a speed of 3 to 5 times. So for the architectures with large scale, PSO-NoC is much better in improving CPU processing time. As a whole, compared with the original algorithm, CPU processing time of PSO-NoC algorithm decreases especially for architectures with large scale.

In summary, PSO-NoC algorithm saves more run time as Mesh size increases. Also, the whole average latency and average throughout of PSO-NoC algorithm improve compared with SA-NoC algorithm. In addition, it can be seen in the figures that PSO-NoC algorithm is much better for large-scale architectures with large number of IP cores.

CONCLUSION

In this paper, we propose an improved algorithm named the particle swarm optimization algorithm to optimize the floorplans (PSO-NoC). The algorithm is based on the advan-

tages of parallel processing units of particle swarm optimization algorithm, and optimizes layout of the tiles to make the floorplanning path and the CPU processing time shorter and more efficient. Also, we adapt and build the original 3D NoC simulator to simulate the proposed algorithm and compare it with the original one. We add calculation of throughput to make the experimental data more comprehensive and persuasive.

In the future, it is helpful for a comprehensive study of architectures to design a graphical interface that can display 3D NoC floorplan in three dimensions. In addition, 3D NoC architecture based on heterogeneous layout we discussed in this paper is from tree graph mapping. So a better mapping algorithm is needed to improve mapping performance of 3D NoC and to ensure the quality of the floorplanning.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (NSFC) (61272006).

REFERENCES

[1] M. Sarrafzadeh, "Transforming an arbitrary floorplan into a sliceable one" In: *Proceedings of the 1993 IEEE/ACM International Conference on Computer-Aided Design*, Santa Clara, CA, USA, 1993, pp. 386-389.

[2] Y. Jin-Tai, L. Kai-Ping, and H. Chun-Tsai, "Sliceable transformation of non-slicing floorplans based on vacant block insertion in LB-packing process", In: *IEEE International 48th Midwest Symposium on Circuits and Systems*, Covington, KY, 2005, pp. 1075-1078.

[3] Y. Jin-Tai, C. Chih-Wei, Y. –F. Luo, and C. Yi-Hsiang, "Packing-driven sliceable transformation for 3D floorplan designs", In: *Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference*, 2008, pp. 85-88.

[4] O. Oluwaseun, "*Parallel implementation of non-slicing floorplan with MPI and OpenMP*", MS thesis, Ryerson University, 2012.

[5] Y. M. Li, "An non-slicing area prejudged algorithm for floorplanning without simulated annealing", In: *2nd International Conference on Computer and Automation Engineering*, Singapore, 2010.

[6] C. Yu-Ning, "Non-slicing floorplanning-based crosstalk reduction on gridless track assignment for a gridless routing system with fast pseudo-tile extraction", In: *ACM International Symposium on Physical Design*, New York, NY, USA, 2008, pp. 134-141.

[7] X. Ning, "Hybrid algorithm for non-slicing floorplans optimization", In: *9th International Conference on Solid-State and Integrated-Circuit Technology*, Beijing, 2008, pp. 2313-2316.

[8] H. –J. Bai, S. –Q. Dong, and X. –L. Hong, "Buffer insertion based on single-pair shortest-path algorithm for interconnect-centric floorplanning", In: *8th International Conference on Solid-State and Integrated Circuit Technology Proceedings*, Shanghai, China, 2006, pp. 1873-1875.

[9] X. Hong, L. Ma, Y. Cai, C. K. Cheng, and J. Gu, "Sequence cloth diagram of Angle module and boundary constraint layout planning algorithm based on angle module expressed in sequence", *Science China*, vol. 32, no. 3, pp. 409-418, 2002.

[10] W. Haiqi, and D. Sheqin, "Topology generation algorithm for application specific network on chip", "*Journal of Computer-Aided Design & Computer Graphics*, vol. 23, no. 9, pp. 1576-1584, 2011.

[11] H. Liang-li , W. Fa-yuan , and W. Feng-jun, "A method based on flock of birds with human-computer technology for packing layout", *Journal of China Academy of Engineering Physics*, vol. 3, pp. 29-31, 2009.

[12] E. F. Y. Young, and T. Ma, "TCG-based multi-bend bus driven floorplanning", In: *Proceedings of the Asia and South Pacific Design Automation Conference*, Seoul, Korea, 2008, pp. 192-197.

[13] E. F. Y. Young, and R. Wang, "3-D floorplanning using labeled tree and dual sequences", In: *Proceedings of the International Symposium on Physical Design*, Seoul, Korea, 2008, pp. 54-59.

[14] J. He, "*The Research and Development of Placement Algorithm for Network on Chip*", PhD thesis, Wuhan University of Technology, 2010.

[15] K. M. Reza, A. Federico, M. Srinivasan, P. Antonio, S. Ciprian, and B. Luca, "A floorplan-aware interactive tool flow for NoC design and synthesis", In: *Proceedings of IEEE International SOC Conference*, Belfast, Northern Ireland, UK, 2009, pp. 379-382.

[16] V. De Paulo, and C. Ababei, "A framework for 2.5D NoC exploration using homogeneous networks over heterogeneous floorplans", In: *International Conference on ReConFigurable Computing and FPGAs*, Cancun, Quintana Roo, Mexico, 2009, pp. 267-272.

[17] J. Kennedy, and R. Eberhart, "Particle swarm optimization", In: *IEEE International Conference on Neural Networks Proceedings*, Perth, WA, Australia, 1995, pp. 1942-1948.

[18] R. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory", In: *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, Nagoya, 1995, pp. 39-43.

[19] P. Hao, and Z. Ming. "Application of immune particle swarm optimizer in neural network training", *Computer Engineering and Applications*, vol. 45, no. 34, pp. 50-52, 2009.

[20] G. Liu, "*Study on Particle Swarm Optimization and its Application in Image Retrieval"* Harbin Engineering University, 2008.

[21] X. Gao, and J. Jiang, "*Research on Particle Swarm Optimization and its Application in Image Retrieval*", Xi'an Electronic Science and Technology University, 2013.

[22] Q. Lin, Y. Zhu, and J. Ye, "Research on projection pursuit classification model based on particle swarm optimization algorithm", *Journal of Changsha Institute of Transportation,* vol. 24, no. 2, pp. 90-95, 2008.

[23] G. Chen, J. Wang, and H. Xie, "Application of particle swarm optimization for solving optimization problem of projection pursuit modeling", *Computer Simulation,* vol. 25, no. 8, pp. 159-161, 2008.

[24] V. de Paulo, and C. Ababei, "3D network-on-chip architectures using homogeneous meshes and heterogeneous floorplans", *International Journal of Reconfigurable Computing*, vol. 2010, pp. 1-12, 2010.

[25] K. Kim, S. Lee, J. –Y. Kim, and M. Kim, "A 125GOPS 583mW network-on-chip based parallel processor with bio-inspired visual attention engine", In: *IEEE International Solid-State Circuits Conference*, San Francisco, CA, 2008.

Received: March 16, 2015 Revised: July 23, 2015 Accepted: July 23, 2015

© Dakun *et al.*; Licensee *Bentham Open.*

This is an open access article licensed under the terms of the (https://creativecommons.org/licenses/by/4.0/legalcode), which permits unrestricted, non-commercial use, distribution and reproduction in any medium, provided the work is properly cited.