An IDPSO Algorithm-based Application Mapping Method for Network-on-Chips

Van-Nam Dinh[a,b], Hung K. Nguyen[a], Minh-Trien Pham[a], Xuan-Tu Tran[a]
[a)] SISLAB, VNU University of Engineering and Technology, 144 Xuan Thuy road, Cau Giay district, Hanoi, Vietnam
[b)] Department of Electronics and Communication, ICTU, TNU, Thai Nguyen, Vietnam.
Corresponding: tutx@vnu.edu.vn

***Abstract*—Application mapping is one of the most challenging issues in designing Network-on-Chips, playing an important role in maximizing the performance of NoC based systems. This paper presents a novel method to map applications onto a targeted NoC architecture using Improved Discrete Particle Swarm Optimization (IDPSO) algorithm. The NOXIM platform has been used to evaluate the efficiency of the proposed method in terms of latency, throughput, and power energy. The obtained results show that the proposed method can help the designers to get better performance for NoC based systems. This information is very useful for designers to decide how they should do in the next steps of designing a real-time system.**

Keywords—Application mapping; Network-on-Chip; Partical Swarm Optimization; NOXIM.

I. INTRODUCTION

Nowadays, designers try to integrate more and more IP (Intellectual Property) cores into a single system to meet the increasingly demand of the applications. This makes the system become much more complex and requires a new on-chip communication solution. Network-on-Chip (NoC) paradigm has become an emerging model for resolving this issue [1]. However, it is not easy to implement the application onto NoC architectures. Then, applications mapping is one of the most vital research aspects that has been attracted by many researchers in the world, especially for complex systems design such as NoCs. Because application mapping techniques will map the functions of an application to the network node, so it can affect to the overall performance of the targeted system.

Researching of application mapping on NoC with the Particle Swarm Optimization (PSO) algorithm states that the optimization problems can improve and even achieve better results in comparing between before and after using this algorithm [2]. There are many researches have proposed application mapping techniques for the NoC based systems such as [3][4][5][6][7].

In [3], the authors provided near-optimal solutions for reducing the runtime significantly. The methodology is using a linear programming (LP) approach followed by a mapping heuristic and this paper achieved a significant decrease in packet latency, but without caring of energy overhead. Dawei Li *et al.* [4] proposed a model that combining between voltage scaling techniques and frequency turning techniques for NoC links to save overall system energy consumption by a directed acyclic graph of application. In [5], a comparison between three real applications is using to evaluate results. Authors have used their experiments to develop a unified communication-aware NoC-based MPSoC mapping and scheduling algorithm. Marcus *et al.* [6] addressed the problem of mapping topologically pre-selected sets IPs buy using multi-objective evolutionary optimization. Paradip Kumar Sahu *et al.* [7] presented a new strategy for Network-on-Chip design by using the approach of Kernighan-Lin bi-partitioning strategy to identify the closeness of cores based on the analyzing their bandwidth requirements. These studies help NoC researchers evaluate the system's performances in terms of latency (cycles), throughput (flits/cycle/IP), and power energy (J) for comparing many algorithms with corresponding to each scenario and application mapping.

In this study, we propose a novel method for mapping applications onto a targeted NoC architecture using an optimization algorithm named as Improved Discrete Particle Swarm Optimization (IDPSO). Then, we also developed a platform based on NOXIM simulator in order to evaluate the system's performance in terms of latency, throughput, and power energy.

The remaining part of this paper is organized as follows. Section 2 introduces briefly the conventional PSO algorithm and then presents the Improved Discrete PSO. Section 3 describes our method to apply the IDPSO to the application mapping for NoC-based systems. The experimental results will be presented in Section 4. Finally, some conclusions and remarks will be included in Section 5.

II. INTRODUCTION TO THE IMPROVED DISCRETE PSO

In 1995, Kennedy and Eberhart proposed a heuristic optimization method as known as the Particle Swarm Optimization (PSO) algorithm [8]. It derives from swarm intelligence (philosophical aspects): fish schooling and bird flocking. With this algorithm, the best solution of the problem can be found quickly thanks to the experiences of each particle and the communication among them in the whole swarm.

978-1-5386-3377-9/17/$31.00 ©2017 IEEE

There are two main factors which can affect to the finding of the best solution: (i) individual factor – the local best position (*local_best*), found by each particle; and (ii) social factor – the global best position (*global_best*), found by the entire swarm. These are similar to the local and global optimization issues in mapping an application to a target system-on-chip platform, especially NoC-based systems. Therefore, the PSO algorithm becomes a potential method in the application mapping of such NoCs. In this section, we will introduce briefly the discrete version of the PSO and its Improved version which is can be used in the application mapping for NoC-based systems.

In fact, the discrete version of PSO is proposed to deal with the application mapping problems [9]. The process of this DPSO algorithm is described as follows:

- **Initialization**

- Generating swarm with both randomly and deterministically (*deterministic initial phase*).
- For each particle,
  +) Evaluate fitness value of each particle
  +) Set local best of each particle
  Find the *global_best* beyond the entire *local_best.*

- **Evolution**

Do

- for each particle p_i :
  +) Identify $SS_i^{global\_best}$ and $SS_i^{local\_best}$;
  +) p_i^{new} = modify p_i by applying $SS_i^{global\_best}$ with probability s_2 and $SS_i^{global\_best}$ with probability s_3 ;
  +) Evaluate *fitness* of p_i^{new};
  +) If this fitness is better than old one, update $local\_best$ for p_i ;
- Find the $global\_best$ beyond the entire $local\_best$;
- Implement multiple stage DPSO;

while the loop condition is still true.

- **Evolution of generations**

Supposing that the position of a particle *i* at k^{th} generation is $p_k^i = \left(p_{k1}^i, p_{k2}^i, p_{k3}^i, ..., p_{kn}^i\right)$ and the corresponding *local_best* and *global_best* are $pbest_k^i$ and $gbest_k^i$. The new position of p_k^i, p_{k+1}^i is calculated as:

$$p_{k+1}^i = \left(s_1 * I \oplus s_2 * \left(p_k^i \to pbest_k^i\right) \oplus s_3 * \left(p_k^i \to gbest_k^i\right)\right) * p_i$$

Where, $a \to b$ implies the swap sequence to transform *a* to *b*. For example, if *a* = <1,2,3,4> and *b* = < 4,1,2,3>, then $a \to b$ = <swap(1,4); swap (2,4); swap(3,4)>. The operator $a \oplus b$ means that the sequence of swap in *a* is followed by the sequence of swap in *b*. The constants s_1, s_2, s_3 are inertia, self-confidence, and swarm confidence values. The $s * a$ means that the swap sequence a will be apply with the probability of s (s = 0 → 1). The variable *i* is the sequence of identity swap.

$SS_i^{local\_test}$ and $SS_i^{global\_best}$ are the swap sequences to align particle p^i with its local best position and global best position. These swaps will be applied with probability of s_2 and s_3. However, these parameters are fixed (constants).

- **Deterministic initial phase**

In the exploration phase, if the local best position is found, the whole swarm is guided to around this position in the exploitation phase. However, because the searching space is very huge, we can get stuck in local optimum in the exploitation phase (note that if we have the *n* IP cores, then we have *n!* possibilities). Therefore, we need to include a set of seeds (*n* particles deterministically generated) to improve quality of solution. The overall process can be described as follows:

- Input: Core Graph (G) and Topology Graph (P)
- Output: Mapped Graph

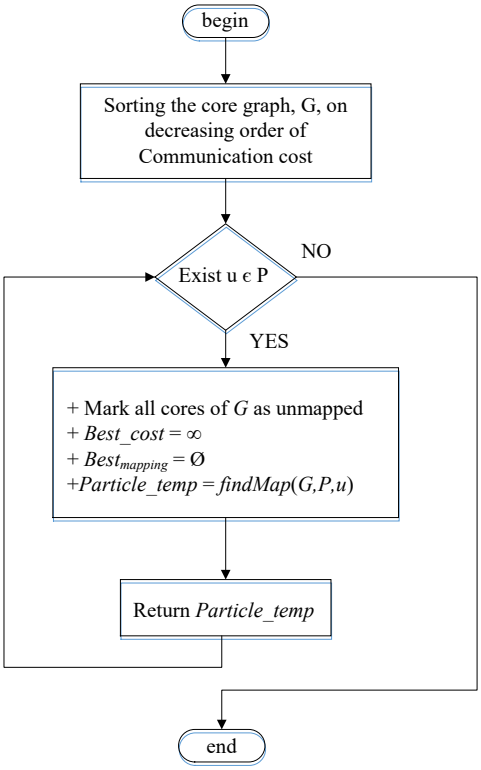

Figure 1: Flowchart of the Deterministic Initial phase.

Where: findMap(G,P,u) can be visualized as (see **Error! Reference source not found.**):

- Inputs:
  - ✓ Core Graph (G), Topology Graph (P).
  - ✓ Cores to be mapped.
  - ✓ Start_Pos (u): the first position of P to be mapped.

- Output: A new particle, *Particle_temp*.

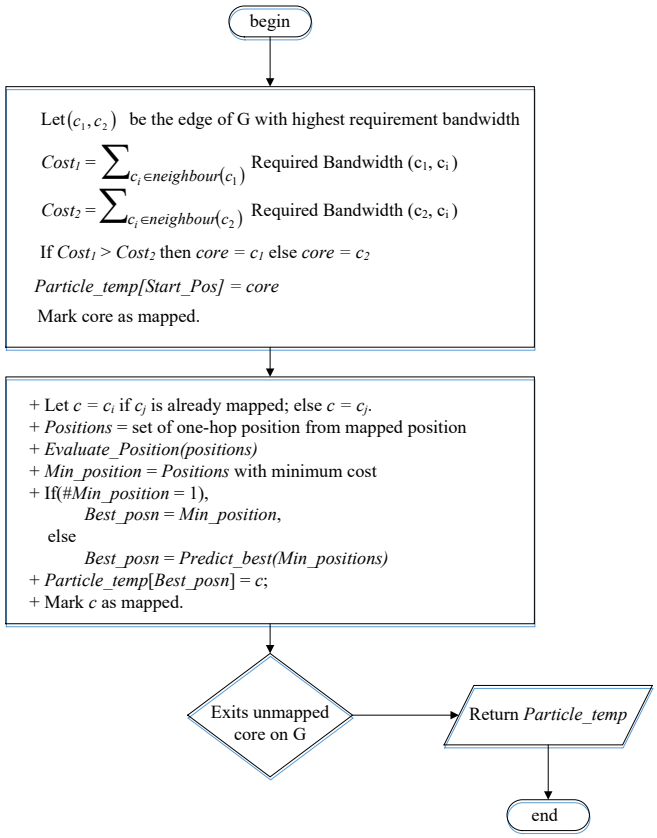

Figure 2: Flow chart of FindMap Function.

Where, *Predict_best* is a function which finds out the better position on P to be mapped a core C. If the number of *Min_positions* of *Predict_best* is more than 2, we will select the first one.

In this work, the Improved DPSO proposed in [10] is used to map an application onto a targeted NoC architecture to get more accurate and stable solution thanks to its cognitive and social learning factors (s_2, s_3). The local factor s_2 and the social factor s_3 are adjusting to change the velocity of each particle.

III. Applying IDPSO to the NoC Application Mapping

To map applications onto a targeted NoC architecture, we proposed a method which is composed of two main parts as shown in Figure 3. Firstly, we use the IDPSO algorithm for finding the best implementation of the application onto the targeted Network-on-Chip. Secondly, the obtained mapped core graph will be sent to NOXIM configuration. NOXIM is open NoC simulation platform which is used to simulate and evaluate network parameters in NoC design [11], [12]. At the second part, the core graph will be translated to network graph and we will see the impact of the IDPSO algorithm on the overall performance of the NoC-based system for the target application.

In this work, the Dual Video Objective Plane Decoder (DVOPD) benchmark has been used as the application which needs to be mapped onto a 2-D mesh NoC architecture. Figure 4 presents the output of first part is the core graph of DVOPD application with the IDPSO algorithm. Then, this core graph will be converted to NoC graph according to the NOXIM. At this stage, we are able to simulate and evaluate the system's performance in terms of delay, data transmission capability, and power consumption. Figure 5 shows the implementation flow for our proposed method.

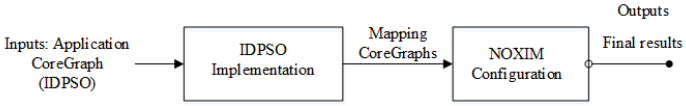

Figure 3: Two main tasks for doing project.

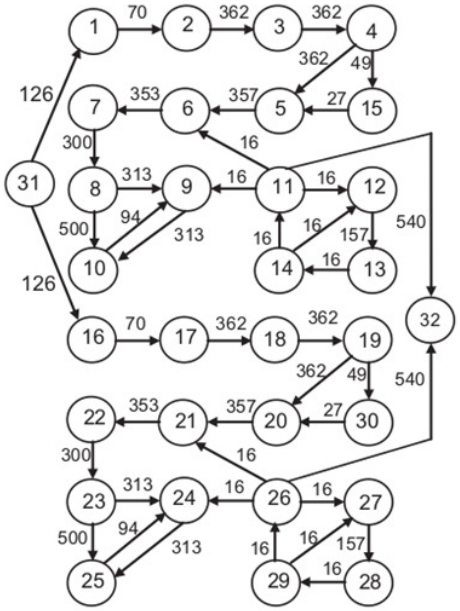

Figure 4: The DVOPD core graph [2].

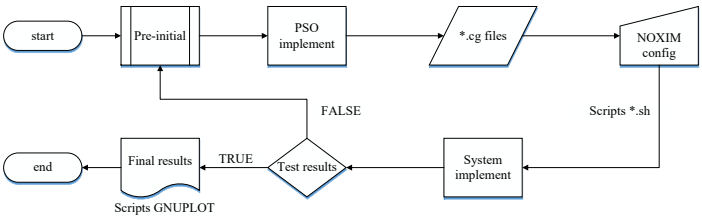

Figure 5: System's step-by-step implementation.

A. Data acquisition

As we presented in the above section, the outputs of the IDPSO algorithm for application mapping are core graphs (*.cg) which includes the positions of cores in the network. According to this core graphs, we apply them for doing in

NOXIM to appraise the method.

Table 1: Data collection of IDPSO algorithm

SBF	DBF	BW	SAF	DAF	Hop counts	SBF
1	2	70	3	2	1	17
2	3	362	2	1	1	18
3	4	362	1	9	1	19
4	5	362	9	17	1	19
4	15	49	9	10	1	20
5	6	357	17	18	1	21
6	7	353	18	19	1	22
7	8	300	19	11	1	23
8	9	313	11	20	2	23
8	10	500	11	12	1	24
9	10	313	20	12	1	25
10	9	94	12	20	1	26
11	9	16	28	20	1	26
11	6	16	28	18	3	26
11	12	16	28	26	2	26
11	32	540	28	29	1	27
12	13	157	26	25	1	28
13	14	16	25	27	2	29
14	11	16	27	28	1	30
15	5	27	10	17	2	31
16	17	70	5	6	1	31

Note:
SBF: Source **Bef**ore IDPSO.
DBF: **D**estination **Bef**ore IDPSO.
SAF: **S**ource **Af**ter IDPSO.
DAF: **D**estination **Af**ter IDPSO.
BW: **B**and**w**idth.

B. Scenarios and configuration

There are four scenarios in this research. All of these scenarios are alternating between two factors that are PIR and packet size; and we have evaluated the outputs based on Latency, throughput and Power Energy of the system.

For the scenario 1, the main idea is changing the Packet Injection Rate (PIR) into the Network, the Packet size in this case will generate in random from (2 Flits in minimum to 10 Flits in maximum). The PIR will change from 0.01 to 0.1 (corresponding to 1% to 100% with 5% jumping in each step of modulation).

For the scenario 2, we will do the simulation with keeping the PIR (in this case we fixed PIR = 0.015) while changing packet sizes {2, 4, 8, 16, 32, 64, 128, 256, 512 and 1028} packets.

For scenario 3, we will reconfigure the system for doing almost steps same to the scenario 1 but choosing the best case of scenario 2 to decide the packet size (choose the best case).

For scenario 4, we will reconfigure the system for doing almost steps same to scenario 2 but choosing the best case of scenario 1 to decide the Packet Injection Rate (choose the best case).

NOXIM has used for simulating and evaluating in this study. Because it allows designers to simulate and evaluate the NoC performance with different network configuration parameters such as network size, buffer size, packet size distribution, packet injection rate, traffic pattern, routing algorithm, traffic time distribution in terms of throughput, delay and power consumption. The obtained results with an 8x8 2D-Mesh NoC architecture is also presented and discussed to demonstrate the NOXIM platform.

Table 2: The configuration of system

Features	*Description*
Network configuration	
Topology	8x8 MESH 2D
Control Flow	Credit Based Mechanism
Routing Algorithm	Deterministic XY algorithm
Switch technique	Wormhole switching
Communication pattern	
HDL	SystemC 2.3.1
Operating System	Linux (Ubuntu 15.10)
Hardware	Intel Core ™ i5 – 2540M CPU @ 2.60Ghz, Architecture i686.

IV. Experimental Results

In this section, we would like to show and analyze the final results in order to demonstrate that the performance of Network on Chip system is improving after using IDPSO according to each scenario.

As having mention before, we will focus on the results in terms of latency (Cycles), throughput (Flits/cycle/IP) and Power energy (J) to evaluate the system's performance.

Figure 6 depicts the final results of the scenario 1 and all cases are combined and taking a comparison in Figure 7. In the scenario 1, the significant points in which include the best case are at 70% (for latency), 85% (for throughput) and 20% (for Power Energy). However, the corresponding worst cases are at 15%, 20% and 40%. The obvious results will be depicted in Figure 7, which used the orange color and the violet color are corresponding to the best and worst cases. In general, almost cases in this scenario have optimized after using IDPSO algorithm because of increasing throughput and both latency and energy are reduced.

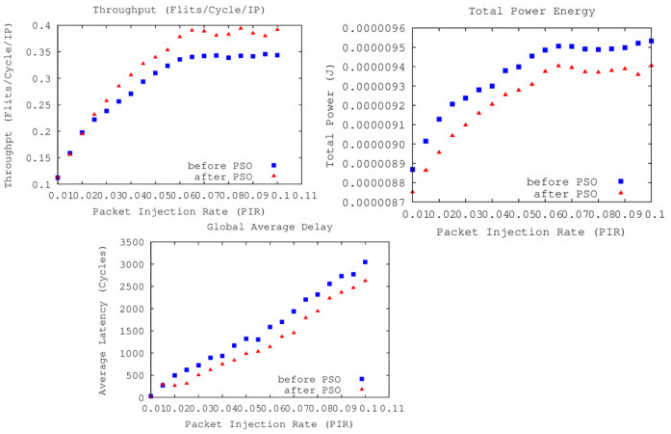

Figure 6: Final results for the scenario 1.

Results for Scenario 1: packetsize from 2 to 10			
PIR	GAD (cycles)	oughput (Flits/cycle	Total Power (e-06 J)
0.01	-9.0415	0.000605	-0.1169
0.015	22.322	-0.002767	-0.1512
0.02	-231.143	-0.00281	-0.17232
0.025	-306.937	0.009355	-0.1636
0.03	-222.019	0.019083	-0.1406
0.035	-272.018	0.028649	-0.12102
0.04	-187.567	0.035326	-0.09482
0.045	-334.955	0.033867	-0.12462
0.05	-336.463	0.02938	-0.12058
0.055	-268.21	0.029896	-0.14673
0.06	-447.77	0.042509	-0.11162
0.065	-334.7	0.049986	-0.10282
0.07	-486.93	0.046709	-0.10968
0.075	-413.25	0.037756	-0.11803
0.08	-374.75	0.044025	-0.11816
0.085	-324.91	0.051825	-0.11277
0.09	-368.68	0.043294	-0.11029
0.095	-304.12	0.03405	-0.16159
0.1	-427.28	0.047972	-0.12716
notes	meaning		
	worst case		
	best case		

Figure 7: Summary of the Scenario 1.

Figure 8 shows the final results of the scenario 2 and all cases also are combined and taking a comparison in Figure 9. For the scenario 2, the worst cases at points which the Packet size equal 2 and 4 packets. And, the best cases including of 32 and 128 points of packet size.

The scenario 2 has achieved same positive points as the scenario 1 because almost cases which after using IDPSO for application mapping are better than before using this algorithm. That means the latency and power energy are decreasing while throughput is increasing.

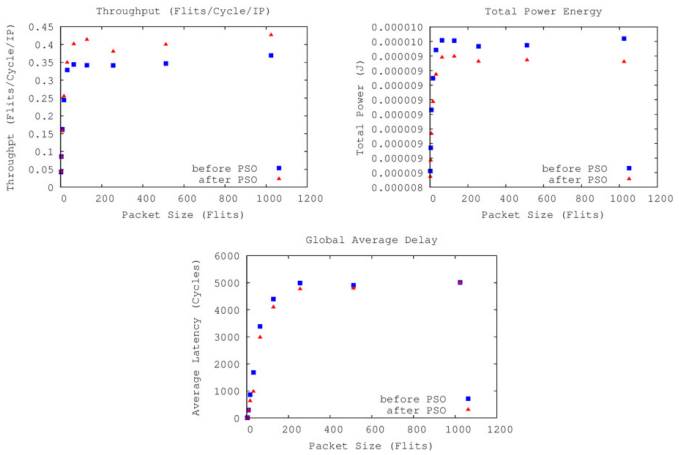

Figure 8: Final results for the scenario 2.

Results for Scenario 2: pir = 0.015			
Packet size	GAD (cycles)	Throughput (Flits/Cycle/IP)	Total Power (e-06 J)
2	-2.0912	0.001466	-0.03839
4	-2.38587	-0.00681	-0.08568
8	-20.555	-0.005559	-0.16307
16	-228.594	0.010835	-0.16192
32	-703.407	0.02067	-0.16723
64	-405.65	0.057284	-0.11701
128	-298.76	0.071776	-0.10808
256	-217.76	0.038993	-0.11043
512	-109.56	0.053389	-0.10155
1024	-14.41	0.056925	-0.15853
notes	meaning		
	worst case		
	best case		

Figure 9: Summary of the Scenario 2.

After analyzing the final results of previous scenarios (S1 and S2), we have proposed the scenario 3 and scenario 4 for extending cases of simulating system.

For the scenario 3, we reconfigure system's parameters as the scenario 1 but changing the packet size with the deterministic point at 32 (packet size = 32). And in the scenario 4, the alternative is to re-configure system's parameter with PIR = 0.085 (85%) instead of 0.01 (1%) as the scenario 2.

Figure 10 and Figure 11 present the final results of the scenario 3 and 4. All cases still improve the performance of system. In the scenario 3, the worst cases of the scenario 1 such as at 15% (latency), 20% (throughput) and 40% (Power Energy) are solved. So, one more advantage of IDPSO has discovered here, which are suitable choosing of parameters during configuration progress can lead to improve systems' performance.

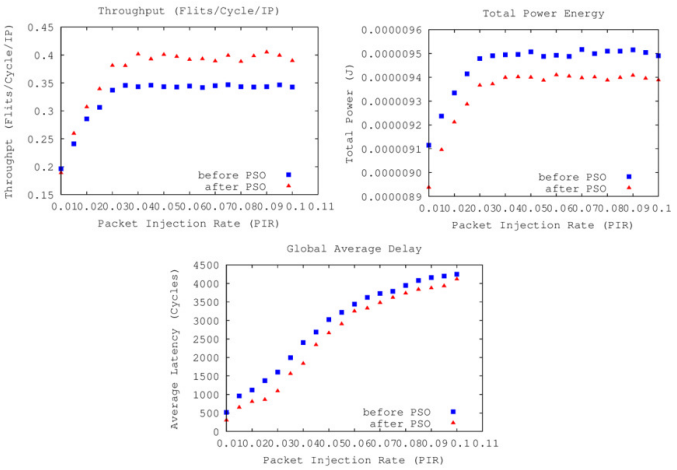

Figure 10: Final results for the scenario 3.

Same to the scenario 3, the worse cases in the scenario 2 (at packet size equal 2 and 4 flits) have solved in the scenario 4 with improving system's performance.

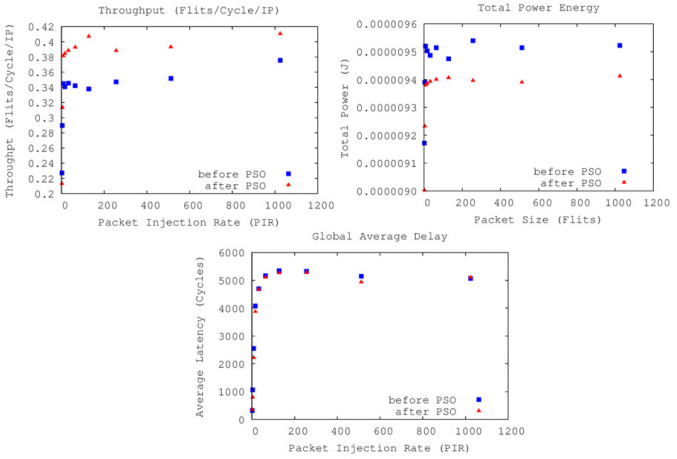

Figure 11: Final results for the scenario 4.

All cases in the scenarios that we have proposed in this study derived very good results after using IDPSO algorithm for DVOPD application. In detail, we can see the latency of the system has been reduced and the throughput increases while the total power and dynamic energy are decreased. All results have positive signs that improving the performance of the NoC system in terms of latency, throughput and power consumption.

This methodology which was used in this study can help us to solve two main things. The first thing is evaluating the performance of NoC in application mapping with DVOPD application based on IDPSO algorithm. Secondly, we also can give more evidence for the reliability of using NOXIM platform simulator.

According to these results, we can announce that the methodology in this study is very useful for researchers who want to use NOXIM to simulate their algorithms on NoC design. There are many other methodologies for doing application mapping research on NoC, but we believe that this is the most one.

V. CONCLUSION

By reconfiguration of NOC system's parameters, we can simulate the various scenarios for demonstrating the results of with IDPSO technique for application mapping issues. This study focuses on evaluating a NoC performance in terms of latency, throughput, and power energy.

There are two significant things that we have achieved in this research. The first thing is we have applied the IDPSO successfully to improve the system's performance. In addition, we have built a flexible system to map and evaluate NoC-based system's performance.

ACKNOWLEDGMENT

This work is supported by the research project No. 102.01-2013.17 (ReSoNoC) granted by Nafosted.

REFERENCES

[1] L. Benini and G. D. Micheli. Networks on Chip: A new SoC Paradigm. IEEE Commut. vol. 35, no. 1, pp. 70-78, Jan, 2002.

[2] Pradip Kumar Sahu, Santanu Chattopadhyay. A Survey on Application Mapping Strategies for Network-on-Chip Design. Journal System of Architecture 59, 2013, pp. 60-76.

[3] Chen-Ling Chou, Radu Marculescu. Contention-Aware Application Mapping for Network-on-Chip Communication Architecture. In Proceedings of the IEEE International Conference on Computer Design (ICCD), 2008.

[4] Dawei Li, Jie Wu. Energy-efficient Contention-aware Application Mapping and Scheduling on NoC-based MPSoCs. Journal of Parallel and Distributed Computing, vol. 96, Oct 2016, pp. 1-11.

[5] Heng Yu, Yajun Ha, Bharadwaj Veeralli. Communication-Aware Application Mapping and Scheduling for NoC-Based MPSoCs. In Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 2010.

[6] Marcus Vinicius Carvalho da Silva, Nadia Nedjah, Luiza de Macedo Mourelle. Optimal Application Mapping on NoC Infrastructure using NSGA-II and MicroGA. In Proceedings of the International Conference on Intelligent Engineering Systems (INES), 16-18 April 2009.

[7] Paradip Kumar Sahu, Nisarg Shah, Kanchan Manna, Santanu Chattopadhyay. A New Application Mapping Algorithm for Mesh based Network-on-Chip Design. In Proceedings of the Annual IEEE India Conference (INDICON), 17-19 Dec. 2010.

[8] I. Kennedy and R. C Eberhart. Particle Swarm Optimization. In Proceedings of the IEEE International

Conference on Neural Network, Nov.-Dec. 1995, pp. 1942-1948.

[9] P. K. Sahu, T. Shah and S. Chattopadhay. Application Mapping onto Mesh-Based Network-on-Chip using Discrete Particle Swarm Optimization. IEEE Transaction on Very Large-Scale Integration (VLSI) System, February 2014, vol. 22, no. 2.

[10] Viet-Huong Nguyen, Giang T.H. Dang, Minh-Trien Pham. Applying Improved Discrete Particle Swarm Optimization onto Mesh-based Network-on-Chip Application Mapping. In Proceedings of the 6th International Conference on Integrated Circuits, Design, and Verification (ICDV), pp. 46-51, 10-11 August 2015, Ho Chi Minh city, Vietnam.

[11] Vincenzo Catania, Andrea Mineo, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. Cycle-Accurate Network on Chip Simulation with NOXIM. ACM Transactions on Modeling and Computer Simulation. Volume 27, Issue 1, November 2016.

[12] NOXIM: Open Source, https://github.com/davidepatti/noxim