Research Article

Effective Task Scheduling and IP Mapping Algorithm for Heterogeneous NoC-Based MPSoC

Peng-Fei Yang and Quan Wang

School of Computer, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Peng-Fei Yang; yangppf@163.com

Received 8 May 2014; Accepted 17 June 2014; Published 10 July 2014

Academic Editor: Yuping Wang

Copyright © 2014 P.-F. Yang and Q. Wang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Quality of task scheduling is critical to define the network communication efficiency and the performance of the entire NoC- (Network-on-Chip-) based MPSoC (multiprocessor System-on-Chip). In this paper, the NoC-based MPSoC design process is favorably divided into two steps, that is, scheduling subtasks to processing elements (PEs) of appropriate type and quantity and then mapping these PEs onto the switching nodes of NoC topology. When the task model is improved so that it reflects better the real intertask relations, optimized particle swarm optimization (PSO) is utilized to achieve the first step with expected less task running and transfer cost as well as the least task execution time. By referring to the topology of NoC and the resultant communication diagram of the first step, the second step is done with the minimal expected network transmission delay as well as less resource consumption and even power consumption. The comparative experiments have shown the preferable resource and power consumption of the algorithm when it is actually adopted in a system design.

1. Introduction

The development of integrated circuit has provided strong support for the integration of multiple processing elements (PEs) in single chip, and the on-chip communication between cores has developed from bus-based approach to two-dimensional and three-dimensional Network-on-Chip (NoC). The network-based highly parallel System-on-Chip (SoC) structure has become the inevitable choice for next generation of complex computer architecture [1]. Nevertheless, the dramatic increase of PEs that can be integrated and the size of executable tasks have brought new problems and challenges to systematic design, among which the dividing and scheduling of the task and IP mapping have become the focus of systematic study.

The NoC-based task scheduling and IP mapping, on the basis of given tasks, type and amount of PEs available, and topology of NoC, assign tasks to suitable PEs, map the PEs to reasonable network topology, improve as much system efficiency as possible while the whole system meets the power consumption, and delay requirements. Its significance includes the following: (1) it serves as the bridge between

applications and architecture and determines the task implementation, processing performance, and efficiency in architecture; (2) as heterogeneous multicore architecture usually associates with particular field, efficient task scheduling could acquire support applications in specific fields; and (3) as the size of tasks and multicore system architecture is increasing, efficient division of mapping will help improve the quality and efficiency of exploring mapping space and thereby improve the performance and efficiency of the entire SoC.

2. Related Work

Current research seldom distinguishes between task scheduling and IP mapping detailedly, and the modeling and analysis is conducted providing that a PE only performs a subtask (in some algorithms, subtasks are simplistic and considered to be PEs). That is to say, the task will be abstracted to a simple form of task model which just gives the calling relationship between subtasks; based on the above information, the scheduling algorithm will allocate as little uptime as possible [2–4]. The approach has many drawbacks: (1) the heterogeneous nature of NoCs and the communication delay

between tasks are usually neglected; (2) as the interdependence among tasks is complex, the model only abstracted the calling relationship between subtasks, with the result that other factors cannot be fully reflected and that transfer costs among different PEs are inadequately considered. The scheduling order designed by these models is not satisfactory in practical operation so that continuous recalculation and adjustment are required during the system operation, which inevitably brings additional burden to the system and poses threats to operating efficiency.

In addition, in terms of the time of scheduling decision, task scheduling can be divided into static scheduling and dynamic scheduling. Static scheduling means that the compiler makes scheduling decision at compiling time, for example, list-based algorithms [5, 6], clustering algorithms [7–9], and duplication-based algorithms [10, 11]. However, static scheduling model has some drawbacks: as the model is an approximation of communication and execution time among processors, it might disagree with the actual implementation of the program or even produce poor scheduling results.

Dynamic scheduling means that a scheduler needs to schedule tasks to appropriate processors for the implementation according to their performance and in a real-time way so that the various requirements for the system can be met. Research in this area mainly employ heuristic algorithm, such as genetic algorithm (GA) [12] and ant-colony-based optimization (ACO) [13, 14] heuristic task scheduling, dynamic scheduling algorithm based on task pool [15], particle swarm optimization (PSO) [16, 17], optimized evolutionary algorithm [18, 19], and dynamic scheduling algorithm based on real-time constraints [20]. Although good scheduling results could be attained when these approaches are applied in task partitioning and mapping, in practice, the inherent defects of these algorithms easily result in many drawbacks during the operation, for example, the convergence speed is slow in the late stage of genetic algorithm; and in the early stage of ant colony algorithm, the inadequate coverage of all collections will lead to disparity between its result and the optimum value; particle swarm optimization is vulnerable to involving local optimization problems.

Meanwhile, in the aspects of NoC topology, through silicon via (TSV) technology [21] and optical interconnection technology [22, 23] have made possible higher IP core density, wider bandwidth, less power consumption, and smaller size on integrated circuit chips. However, the resource occupancy and power consumption brought by NoC must be considered. In order to decline the NoC occupancy of limited resource and further decrease power consumption, various kinds of heterogeneous NoC topology are designed [24–26] to suit differentiated needs for network transmission delay and bandwidth of different types of PEs. Currently, most algorithms have not taken the effect of heterogeneous topology on system performance into consideration. If PEs of different types, in the premise of balanced power consumption, are mapped to reasonable area according to performance requirement and data transmission delay are minimized, the performance of system could be greatly improved.

Based on the analysis above, the whole design process is divided into two stages. As shown in Figure 1, the first

stage is task dividing and scheduling. When the improved task model could faithfully reflect the real intertask relation, the local optimum question of particle swarm algorithm is solved and the optimized PSO algorithm is used to divide a big task into proper granular-sized small tasks featuring high cohesion and low coupling according to traffic and calling relationship. There exists high parallelism among these small tasks. Then assign these small tasks to corresponding PE according to the task nature and generate communication diagram to achieve the first step with expected less transfer cost as well as the least task execution time. Then the process comes to the IP mapping stage. In this stage, by referring to communication diagram and the performance disparity and delay information of topology of NoC, the PEs are reasonably mapped into switching node of NoC so as to achieve least network transmission delay with less resource occupancy and even power consumption and less resource pieces so that the system performance could avoid fluctuation when new tasks need scheduling.

The rest of the paper is organized as follows. Section 3 shows the detailed description of task dividing and scheduling. Section 4 illustrates the process of IP mapping. A comparative experiment result is shown in Section 5. Section 6 concludes the paper.

3. Task Dividing and Scheduling

Although the types and quantities of PEs integrated in heterogeneous multicore system based on NoC are expanding, the size of application task varies and the current task scheduling algorithm often assign and map the task in accordance with the numbers of utilizable PEs, which, to some tasks of small size, may result into problems; on one hand, as the tasks are divided into subtasks of extremely small size, communications among subtasks would become overfrequent which may lead to prolonged task execution time; on the other hand, inadequate utilization of the performance of PEs may result into increased system power consumption and reduce overall system efficiency.

This paper superimposes tasks on a PE until the computing resource of the PE is occupied at an appropriate ratio (settings are based on the performance requirement of system as well as PEs), and then new PEs are added. The approach not only ensures that tasks are divided into subtasks of appropriate size but also ensures that every PE invoked is efficiently used, thus bringing the best overall performance.

3.1. Task Model. A task could be divided into N subtasks among which there exists certain execution sequence or control logic and these subtasks are processed by M (m types, $m \leq M$) PEs. Assuming that the processing time of m types of PEs for every subtask, communication overhead among PEs, and amount of data transmission among interdependent subtasks are known, the task on heterogeneous multicore can be abstracted into a quintuple:

$$\text{DAG} = (V, E, \text{Type}, \text{PCU}, C), \quad (1)$$

FIGURE 1: Two stages of task scheduling and IP mapping.

- (1) V : task node-set in DAG application; that is, the vertex $v \in V$ means that v is a subtask in V . And the number of subtasks in DAG application is N .
- (2) E : the frontier set in DAG application; that is, $e_{i,j} \in E$ means that there exists data communication between v_i and v_j ; the direction of arrow indicates the direction of data transmission.
- (3) Type (v): the type of the task. For instance, we can use 1, 2, 3, ... to represent different computing types. In addition, the type-set of tasks corresponds with that of PEs, which means that a task could only be scheduled to PE matching its type. This could be expressed by the matrix $D = d_{i,j}$, where the lines represent the tasks, the columns represent the PEs, element $d_{i,j} = \infty$ represents task v_i which cannot be executed in P_j and $d_{i,j} = a$ represents task v_i which can be executed in P_j with the execution time of a .
- (4) PCU: the running cost of every type of PE per unit time, in which element PCU_r ($1 < r < m$) represents the running cost of r th type of PE per unit time.
- (5) C : the collection of the communication overhead of directed edge. $C_{i,j}$ represents the transfer cost of subtasks v_i and v_j when they pass the directed edge $e_{i,j}$. When v_i and v_j are scheduled to the same PE, $C_{i,j}$ equals zero.

The target of task dividing and scheduling is to find a proper strategy of assigning and scheduling while meeting task processing sequence and resource limitation which could assign N subtasks to PEs with proper amount and schedule the execution order of every subtask in a reasonable manner, thus achieving minimum completion time of overall task with every task suiting the dependency graph. Based on task model, an improved particle swarm algorithm is used to conduct computation.

3.2. Coding and Decoding. The resource occupation of every subtask is encoded by indirect encoding. The encoding length depends on the amount of subtasks. Every particle corresponds to a certain task assigning strategy.

Assume there exists N subtasks which are encoded by sequential encoding in a task and M PEs available which are classified into m types. For example, when $N = 10$, $m = 3$, particle (3, 2, 1, 1, 3, 2, 1, 2, 3, 3) is a feasible scheduling

scheme; the particle is encoded as shown in Table 1, and as shown in Table 2, by decoding the particle, we can acquire the assigning condition of subtasks in every type of PE. Then, as shown in Table 3, after assigning the subtasks, PEs of reasonable amount are assigned to every type of PE in accordance with the processing ability and the total amount of tasks to be processed.

It follows from the task model that the running time of every subtask in different PEs is already known. The running time on every type of PE is defined as

$$\text{Sub\_TFT} = \sum_{i=1}^n T_{i,r}. \quad (2)$$

$T_{i,r}$ represents the running time of subtask i on the r th type of PE, and n represents the amount of subtasks assigned to r th type of PE. The execution time of the entire task is obtained as follows:

$$\text{TFT} = \text{Max}_{r=1}^k \{ \text{Sub\_TFT}_r \}. \quad (3)$$

The overall operation cost is given as

$$\text{Run\_Cost} = \sum_{r=1}^k \text{Sub\_TFT}_r \cdot \text{PCU}_r. \quad (4)$$

Assuming that the task set in the m th type of PE is V_m and the task set assigned to n th type of PE is V_n , the transfer cost between PE_m and PE_n is defined as

$$\text{Tran\_Cost}_{mn} = \sum_{\forall i,j} C_{i,j}, \quad (v_i \in V_m, v_j \in V_n). \quad (5)$$

The overall transfer cost is obtained as follows:

$$\text{Tran\_Cost} = \sum_{\forall m \neq n} \text{Tran\_Cost}_{mn}. \quad (6)$$

3.3. Initialization and Fitness Function. Assuming that the population size is s , amount of subtasks is N , and amount of types of PEs is m , the description of initialization of the population can be as follows: among the randomly generated s particles, the position of i th particle is represented by vector $x_i = (x_{i1}, x_{i2}, \dots, x_{im})$, ($1 \leq i \leq s, 1 \leq n \leq N$), in which x_{ij} ($1 \leq x_{ij} \leq m$) represents that, in the i th particle, task j is assigned to PE of x_{ij} type for operation; velocity is represented

TABLE 1: Example of particle coding.

Subtask number	1	2	3	4	5	6	7	8	9	10
Type of PE	3	2	1	1	3	2	1	2	3	3

TABLE 2: Example of decoding.

Type of PE	Subtask number
1	3, 4, 7
2	2, 6, 8
3	1, 5, 9, 10

TABLE 3: Task dividing.

Type of PE	Number of PE	Subtask number
1	1	3, 4, 7
2	2	2, 6, 8
3	3	1, 5
3	4	9, 10

by vector $v_i = (v_{i1}, v_{i2}, \dots, v_{in})$, ($1 \leq i \leq s, 1 \leq n \leq N$), in which $-m \leq v_{ij} \leq m$.

The fitness function of time is defined as

$$\text{Fit\_Time}(i) = \frac{1}{\text{TFT}_i}, \quad (1 \leq i \leq s), \quad (7)$$

where TFT_i represents the overall completion time of the i th particle; the fitness function of cost is obtained as follows:

$$\text{Fit\_Cost}(i) = \frac{1}{\text{Run\_Cost}_i + \text{Tran\_Cost}_i}, \quad (1 \leq i \leq s). \quad (8)$$

The overall fitness function is obtained as follows:

$$\text{Fitness} = \text{Fit\_Time}(i) + \text{Fit\_Cost}(i). \quad (9)$$

The algorithm will select particles with higher fitness value so that it could provide excellent basis for generating excellent particles of the next generation.

3.4. Position and Velocity Updating. In every iteration, the particle would update its velocity and position by (10) in accordance with its optimal historical position and the optimal position of the population. Only when the current position has better adaptive value comparing to its historical optimal position would the historical position be replaced by the current position

$$\begin{aligned} v_{id}^{k+1} &= w_k \cdot v_i^k + c_1 \cdot r_1 \cdot (P\_best_i - x_i^k) \\ &\quad + c_2 \cdot r_2 \cdot (G\_best_i - x_i^k), \\ x_i^{k+1} &= x_i^k + v_i^k. \end{aligned} \quad (10)$$

P_best_i is the best position experienced by i th particle, G_best_i is the best position experienced by all particles in the population, w_k is significant for balancing the algorithms

capability of global and local searching, and the paper adopts the decreasing inertia weight as follows:

$$w_k = \frac{w_{\text{start}}(w_{\text{start}} - w_{\text{end}})(\text{Gen} - k)}{\text{Gen}}. \quad (11)$$

w_{start} and w_{end} represent, respectively, the initial inertia weight and the inertia weight when maximum iteration times Gen is reached; k is the current iterations. By adopting the inertia weight above, an algorithm with strong global search capability in the early stage of iteration and more accurate local search capability in the late stage can be gotten.

3.5. Flow of Algorithm

- (1) Randomly initialize the position and velocity of the particle swarm based on the description in "Initialization and Fitness Function."
- (2) Compute the velocity and position of every particle.
- (3) Compute the fitness value of every particle and set P_best_i and G_best_i .
- (4) If P_best_i and G_best_i remain unchanged after many iterations or the algorithm reached maximum iterations, output the optimum solution, end the algorithm, and go to step 6.
- (5) Go to step 2.
- (6) Assign PEs of reasonable amount to every type of PE in accordance with the processing ability and total amount of tasks to be processed.

4. IP Mapping

After task dividing and scheduling, the IP communication diagram is formed. In the multicore system based on NoC, the further need is how to reasonably map these PEs into NoC nodes and minimize the network transmission delay during the task execution under conditions that the resources are less occupied and energy consumption is balanced. This is the question of IP mapping.

There are often two orientations in IP mapping: either to minimize the internal communication cost or to minimize the external communication cost [27, 28]. Both orientations have their pros and cons; the former might lead to increased competition among external resources and add more computation overhead later in mapping when increasing use ratio of system resource; the later tends to arrange surplus resources well and successfully decreases competition of external resources with little changes in computation overhead. However, as each local mapping area is incomplete, it produces only second-best mapping solutions, thus undermining the global mapping optimization. While designing

an IP mapping algorithm, it is necessary to make a careful balance between the two orientations above.

In the meantime, as described above, PEs of different types would have different requirements on a NoC communication capability. In order to save on-chip resource and decrease system consumption, various heterogeneous network topologies are designed. Therefore, during IP mapping, the matching between the communication requirements and on-chip communication capability entails comprehensive consideration.

The paper, based on the property of PEs to be mapped and the characteristics of distribution of transmission capability on topology, maps the PEs of high communication requirement to high-capability area, balances communication cost internal with that external, and achieves on-chip communication of system by minimum transmission delay and less resource occupancy. The mapping algorithm consists of two parts: the expression of the network topology by two-dimensional matrix and the IP mapping. They are detailed as follows.

4.1. IP Communication Diagram and NoC Topology. The communication diagram can be abstracted into a triple CDAG = (P, E, C), where

- (1) P represents the set of PEs in the communication diagram; that is, $p_i \in P$ is a PE with execution task;
- (2) E represents frontier set in DAG application; that is, $e_{ij} \in E$ indicates that there exists data exchange between p_i and p_j ;
- (3) C represents communication cost in undirected edge and C_{ij} represents the total communication data between p_i and p_j .

It is complicated to express NoC topology directly, especially, three-dimensional NoC. Nevertheless, twodimensional matrix expresses topology well and many properties of matrix could also be applied to topology computation. Therefore, the paper expresses topology by two-dimensional matrix before IP mapping.

Three-dimensional mesh topology can be taken as an example. Shown in Figure 2(a) is a $4 \times 4 \times 2$ three-dimensional NoC topology; the red vertices represent bottom switching nodes and the black ones represent upper switching nodes. Figure 2(b) is its two-dimensional expansion diagram, by which we can be free of the complexity in studying the three-dimensional topology. For the convenience of expression and computation, the position of nodes in expansion diagram is expressed by matrix. The position of nodes in Figure 2(b) can be seen in Figure 2(c). There may exist areas where communication transmission capability is higher than that of others to fulfill the higher communication requirement of some PEs; as shown in Figure 2(c), the green areas represent areas in which there exist switching nodes with higher communication performance. For the integrity of matrix expression, areas without switching nodes are filled with shadow; in the later computing, nodes in these areas are assumed to be assigned out already.

Through the approach above, there forms one-to-one correspondence between the position of every node in three-dimensional NoC topology and that of every element in matrix. IP mapping conducts computing optimization on the basis of matrix.

4.2. IP Mapping. Before introducing the concrete algorithm, three parameters are given as follows.

Definition 1. Manhattan Distance MD(i, j): in a plane, the Manhattan Distance between point $P_i(x_1, y_1)$ and $P_j(x_2, y_2)$ is defined as

$$MD(i, j) = |x_1 - x_2| + |y_1 - y_2|. \quad (12)$$

Definition 2. Euclidean Distance ED(i, j): in a plane, the Euclidean Distance between point $P_i(x_1, y_1)$ and $P_j(x_2, y_2)$ is defined as

$$ED(i, j) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \quad (13)$$

Definition 3. Communication cost in mapped area is obtained as follows:

$$\text{Com\_cost} = \sum_{\forall C_{ij} \in C} C_{ij} \cdot MD(L(p_i), L(p_j)), \quad (14)$$

in which C_{ij} represents the total communication traffic between P_i and P_j in communication diagram and $MD(L(p_i), L(p_j))$ represents Manhattan Distance of mapped position on topology between P_i and P_j .

The target of the algorithm is to map PEs with high communication requirement to topology area with high communication capability and find out a mapping scheme which has minimum Com_cost in the results.

The algorithm divides communication diagram into collections H and L according to whether or not included PEs need to be mapped in area with high capability. In the collection $H = \{h_1, h_2, \dots, h_i\}$ with high communication requirement, the sequence is $|h_1| \geq |h_2| \geq \dots \geq |h_i|$ according to the amount of PEs with high communication requirement; in the collection $L = \{l_1, l_2, \dots, l_i\}$ without high communication requirement, the sequence is $|l_1| \geq |l_2| \geq \dots \geq |l_i|$ according to amount of PEs contained. The execution steps of mapping algorithm are as follows.

- (1) Start mapping computation from collection h_1 , choose communication area with high communication capability which could contain the minimum set of PEs with high communication requirement in h_1 on topology as the beginning area of mapping. Name the mapped PEs as assigned area and name the occupied switching nodes area on topology as mapped area.
- (2) Start from the PE with maximum communication traffic (sum of input and output) and map it to the switching node in the area of high communication capability whose available neighboring nodes number is nearest to PE node degree.

FIGURE 2: Topology and its expression by matrix.

FIGURE 3: Description of mapping process.

FIGURE 4: Comparison of algorithm velocity.

- (3) Choose the node which has maximum communication data with assigned area as the next PE to be mapped.
- (4) Correspond the PE to switching node which has minimum Manhattan Distance with mapped area. If more than one node meet requirement, choose the node whose available neighboring nodes number is nearest to PE node degree; if there are still more than one node, then choose the switching node which has minimum Euclidean Distance from the center of mapped area.

- (5) Repeat step 3 and step 4 until all PEs are mapped and start algorithm of another PE diagram to be mapped.

Figure 3 is the simple description of mapping process. In IP communication diagram, the red PEs represent PEs with high communication requirement and blue area represents assigned area; in the topology the green area represents area of switching nodes with high communication capability and area encircled by red line represents mapped area.

The mapping algorithm arranges PEs with direct communication relationship to neighboring nodes, ensuring the road between source node and destination node to be shortest without any conflicts with other transmission roads, thus minimizing the delay in the whole mapping area.

5. Experiment and Simulation

The comparison and evaluation on the performance of designed algorithm are given from two aspects. The first one is the velocity efficiency itself of task dividing and scheduling algorithm. By computing tasks of the same size according to GA, ACO, PSO, and algorithm in this paper, respectively, and comparing the running time, we can prove the efficiency of algorithm. This part is conducted in Matlab with iterations being 200 times; the comparison of time required for running algorithms is shown in Figure 4.

FIGURE 5: Comparison of mapping effect.

The other one is the comparison on actual mapping effect (Figure 5). By comparing the operation of different scheduling results from the above algorithms in NoC simulation environment and computing the delay of power consumption of system, respectively, we can prove the superiority of the algorithm of this paper in scheduling.

6. Conclusion

In this paper, the task scheduling model is further improved and the operating cost per time unit is employed as uniform measurement for PEs of different types and simplifies algorithm; task dividing and scheduling and IP mapping are handled separately so that the resultant algorithm scheduling is more efficient and truthful. The target of scheduling not only considers the total time spent but also considers the time cost and resource cost during the task running so as to achieve comprehensive optimization of system performance.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] C. Addo-Quaye, "Thermal-aware mapping and placement for 3-D NoC designs," in *Proceedings of the IEEE International SOC Conference*, pp. 25–28, September 2005.
- [2] A. K. Singh, W. Jigang, A. Prakash, and T. Srikanthan, "Mapping algorithms for NoC-based heterogeneous MPSoC platforms," in *Proceedings of the 12th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD '09)*, pp. 133–140, August 2009.
- [3] K. Ganeshpure and S. Kundu, "On runtime task graph extraction in MPSoC," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pp. 171–176, IEEE, 2013.
- [4] Y. Z. Tei, M. N. Marsono, N. Shaikh-Husin, and Y. W. Hau, "Network partitioning and GA heuristic crossover for NoC application mapping," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '13)*, pp. 1228–1231, Beijing, China, May 2013.
- [5] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.
- [6] M. I. Daoud and N. Kharma, "Efficient compile-time task scheduling for heterogeneous distributed computing systems," in *Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS '06)*, vol. 1, pp. 11–19, IEEE, Minneapolis, Minnesota, July 2006.
- [7] M. Wu and D. D. Gajski, "Hypertool: a programming aid for message-passing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 3, pp. 330–343, 1990.
- [8] T. Yang and A. Gerasoulis, "DSC: scheduling parallel tasks on an unbounded number of processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 9, pp. 951–967, 1994.
- [9] S. J. Kim and J. C. Browne, "A general approach to mapping of parallel computation upon multiprocessor architectures," in *Proceedings of the International Conference on Parallel Processing*, vol. 2, pp. 1–8, 1988.
- [10] Y.-C. Chung and S. Ranka, "Applications and performance analysis of a compile-time optimization approach for list scheduling algorithms on distributed memory multiprocessors," in *Supercomputing*, pp. 512–521, 1992.
- [11] I. Ahmad and Y. Kwok, "A new approach to scheduling parallel programs using task duplication," in *Proceedings of the International Conference on Parallel Processing*, vol. 2, pp. 47–51, 1994.
- [12] M. Sayuti and L. S. Indrusiak, "Real-time low-power task mapping in networks-on-chip," in *Proceedings of the IEEE*

- Computer Society Annual Symposium on VLSI (ISVLSI '13)*, pp. 14–19, 2013.
- [13] F. Ferrandi, P. L. Lanzi, C. Pilato, D. Sciuto, and A. Tumeo, “Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 6, pp. 911–924, 2010.
  - [14] L. S. Junior, N. Nedjah, and L. de Macedo Mourelle, “CO approach in static routing for network-on-chips with 3D mesh topology,” in *Proceedings of the IEEE Fourth Latin American Symposium on Circuits and Systems (LASCAS '13)*, pp. 1–4, IEEE, Cusco, Peru, February 2013.
  - [15] R. Hoffmann, A. Prell, and T. Rauber, “Dynamic task scheduling and load balancing on cell processors,” in *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP '10)*, pp. 205–212, February 2010.
  - [16] M. B. Abdelhalim, “Task assignment for heterogeneous multiprocessors using re-excited particle swarm optimization,” in *Proceedings of the International Conference on Computer and Electrical Engineering (ICCEE '08)*, pp. 23–27, Phuket, Thailand, December 2008.
  - [17] M. S. Sidhu, P. Thulasiraman, and R. K. Thulasiram, “A load-rebalance PSO heuristic for task matching in heterogeneous computing systems,” in *Proceedings of the IEEE Symposium on Swarm Intelligence (SIS '13)*, pp. 180–187, IEEE, Singapore, April 2013.
  - [18] Y. Wang and C. Dang, “An evolutionary algorithm for global optimization based on level-set evolution and latin squares,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 579–595, 2007.
  - [19] Y.-P. Wang, Y.-C. Jiao, and H. Li, “An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme,” *IEEE Transactions on Systems, Man and Cybernetics C: Applications and Reviews*, vol. 35, no. 2, pp. 221–232, 2005.
  - [20] O. Arnold and G. Fettweis, “Power aware heterogeneous MPSoC with dynamic task scheduling and increased data locality for multiple applications,” in *Proceedings of the International Conference on Embedded Computer Systems (SAMOS '10)*, pp. 110–117, 2010.
  - [21] G. De Micheli and L. Benini, *Networks on Chips: Technology and Tools*, Academic Press, 2006.
  - [22] D. A. B. Miller, “Rationale and challenges for optical interconnects to electronic chips,” *Proceedings of the IEEE*, vol. 88, no. 6, pp. 728–749, 2000.
  - [23] D. A. B. Miller, “Device requirements for optical interconnects to silicon chips,” *Proceedings of the IEEE*, vol. 97, no. 7, pp. 1166–1185, 2009.
  - [24] M. O. Agyeman and A. Ahmadiania, “Optimising heterogeneous 3D networks-on-chip,” in *Proceedings of the 6th IEEE International Symposium on Parallel Computing in Electrical Engineering (PARELEC '11)*, pp. 25–30, April 2011.
  - [25] Y. Ye, J. Xu, X. Wu, W. Zhang, W. Liu, and M. Nikdast, “A torus-based hierarchical optical-electronic network-on-chip for multiprocessor system-on-chip,” *ACM Journal on Emerging Technologies in Computing Systems*, vol. 8, no. 1, article 5, 2012.
  - [26] H. A. Khouzani, S. Koochi, and S. Hessabi, “Fully contention-free optical NoC based on wavelength routing,” in *Proceedings of the 16th CSI International Symposium on Computer Architecture and Digital Systems (CADS '12)*, pp. 81–86, May 2012.
  - [27] C. Chou and R. Marculescu, “User-aware dynamic task allocation in networks-on-chip,” in *Proceedings of the Design, Automation and Test in Europe (DATE '08)*, vol. 1–3, pp. 1074–1079, March 2008.
  - [28] C. Chou and R. Marculescu, “Run-time task allocation considering user behavior in embedded multiprocessor networks-on-chip,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 1, pp. 78–91, 2010.