Heuristics for Dynamic Mapping of Quality Adjustable Applications on NoC-based Reconfigurable Platforms

Nguyen Van Cuong[1,2], Le Dinh Tuyen[1], Dao Vu Tuan[1], Tran Thanh Hai[1], Pham Ngoc Nam[1]*
[1]*Hanoi University of Science and Technology – No. 1, Dai Co Viet Str., Hai Ba Trung, Ha Noi, Viet Nam*
[2]*Industrial University of Ho Chi Minh City, No. 12 Nguyen Van Bao, Go Vap District, Ho Chi Minh, Viet Nam*
Received: June 06, 2016; accepted: June 9, 2017

Abstract

Network on Chip and FPGA-based reconfigurable Systems on Chip are a new trend to provide high performance, flexibility, reducing cost and time to market for the embedded systems.The problem of mapping quality adjustable applications onto heterogeneous NoC-based reconfigurable platforms at run-time with resource constraints while ensuring the maximum overall quality of service of the applications is a big challenge. In this paper, an efficient mapping technique is proposed to solve this problem which consists of a near convex region selection strategy and a dynamic heuristic mapping algorithm. Simulation results show that the proposed technique is very flexible and more than 43% average overall QoS can be achieved compared to some existing solutions. Besides, this technique allows new applications to be easily added to the system in the future.

Keywords: Network on Chip, dynamic mapping, reconfigurable region, quality level, heterogeneous

1. Introduction

For cost-effective reason, embedded systems are often designed to accommodate multiple applications with different Quality of Service (QoS) and processing requirements. With the recent advancements in FPGA technology [1], System on Chip (SoC) FPGAs have become promising platforms for high performance embedded systems because they provide a good balance between performance, rapid time to market, cost, and flexibility. Following this trend, a number of FPGA based embedded systems have been developed to support multimedia and signal processing applications [2-6]. These applications often require high-performance communication infrastructure and data processing capability.

In order to provide a high performance communication infrastructure that connects different processing elements (PE) of a complex SoC, Network on Chip (NoC) has been proposed as an alternative to the traditional bus and point-to-point connection [7-8]. In addition, a heterogeneous reconfigurable platform with the flexibility of embedded processors and the computation efficiency of some NoC based reconfigurable regions (RRs) has exhibited a number of advantages over homogeneous platforms [9-10]. In such a platform, the embedded processors are typically used to implement management and low complexity tasks while the NoC based reconfigurable FPGA fabrics are used to accelerate computational intensive tasks. The reconfigurability aspect allows FPGA platforms to adapt themselves to the various processing requirement of applications. On the other hand, many applications are designed in such a way that their quality level can be adjusted to match the processing capability of hardware platforms. Therefore, research of mapping problem for quality adjustable applications onto heterogeneous NoC-based reconfigurable platforms at run-time is a new trend to provide high performance and flexibility for the embedded systems.

In our previous study [11], we formulated the problem of mapping quality adjustable applications onto heterogeneous NoC-based reconfigurable platforms at run-time under resource constraints while ensuring the maximum overall QoS of the applications and proposed an algorithm to find the optimal solution for the problem. However, this algorithm can only be applied for a few applications with small number of quality levels.

In this paper, we propose an efficient solution to solve the problem of mapping presented in [11] for many applications with different size. The proposed solution consists of a new near convex region selection strategy and a dynamic heuristic mapping algorithm. First, a near convex region that can accommodate a number of application tasks is found, then the application tasks are mapped into the selected near convex region. Besides, this technique allows new applications to be added easily to the system in the future.

* Corresponding author: Tel.: (+84) 983608425
Email: nam.phamngoc@hust.edu.vn

A number of solutions have been proposed recently for the problem of selecting region and dynamic mapping at run time of applications on NoC-based platforms. L. Ost et al. [12] introduce a unified model framework for task mapping on heterogeneous NoC-based platforms at run-time. This study considers several different platforms and the characteristics of the applications, then maps the applications on different soft-core processors on FPGA to optimize some parameters such as energy consumption, delay and communication costs. In addition, this study also makes a trade-off between area cost of the platform and application execution time to achieve the overall performance for the system. In [13], the authors propose a communication-aware run-time heuristic for mapping multi-applications onto NoC-based MPSoC platform. This solution examines the available resources and maps the communicating tasks on the nearest PE in the order of left, down, top and right. By examining communications with the previously mapped tasks on the target processing element, tasks are mapped.

Chen. L. Chou and R. Marculescu [14-15] propose a region selection algorithm and a heuristic for run-time application mapping onto NoC platforms. This algorithm first chooses a near convex region and then maps the applications to the selected region to optimize communication energy consumption. In the most recent work [16], Haghbayan et al. propose a runtime mapping technique named MapPro which uses a proactive choice strategy for adding new applications into the system and hill climbing algorithm to select the first node after conducting mapping applications in selected regions to optimize energy consumption, latency, and congestion. It should be noted that in all the aforementioned studies, the quality of mapped applications is fixed and cannot be adjusted during application execution.

The remainder of the paper is organized as follows. Section 2 presents system models. Section 3 describes run-time mapping problem and heuristic algorithms. Simulation results and discussion are provided in Section 4. Finally, conclusions and future work are given in Section 5.

2. Preliminaries

2.1. Application model

In this paper, we consider quality adjustable applications that have a fixed task graph and the quality of the applications can be adjusted by changing the amount of data to be processed by the task graph.

2.1.1. Application task graph

Definition 1: An Application Task Graph is represented as an acyclic directed graph $ATG = A(V, E)$, where V is the set of tasks and E is the set of all edges, each of which connects between two tasks as shown in Figure 1. Each node $v_k \in V$ is a task with attributes ($t_{\text{id}}, t_{\text{type}}, t_{\text{comp}}, t_{\text{reqcomp}}$). Where, t_{id} is identification parameter; t_{type} is the task type (t_{HW}: hardware or t_{SW}: software) with hardware tasks are generated from an HDL language and software tasks are created from high level programming languages such as C/C++; t_{comp} is the computation time of the task on the hardware or software resource. A task that is executed on the hardware resource will require smaller time than on the software resource [17]; t_{reqcomp} is the minimum required time to complete the task or deadline time.

Each edge e_{rs} represents the relationship between task v_r and v_s. It has attributes ($t_{\text{comm}}, t_{\text{reqcomm}}$). Where t_{comm} is communication time from task v_r to v_s; t_{reqcomm} is the lowest communication time requirement from task v_r to v_s.

2.1.2. Quality model

Consider an application App_i that has N_i quality levels $Q_{i1}, Q_{i2}, Q_{i3}, ..., Q_{iN_i}$. Each quality level requires a certain set of resources including computation, communication, area, and power resources. The higher the quality level is, the more resource is required. For example, suppose $Q_{11} > Q_{12} > Q_{13} > \ldots > Q_{1N_1}$ the time needed to run the application at each quality level will be $t_{11} > t_{12} > t_{13} > \ldots > t_{1N_1}$. Each quality level Q_{ij} provides the user a perceptual quality, which can be represented by a benefit value B_{ij}. We adopt the model that represents the relationship between the benefit value and quality level of the application as presented in [17]. The higher the quality level is, the bigger the benefit becomes, i.e., $B_{i1} > B_{i2} > B_{i3} > \ldots > B_{iN_i}$ $(0 < B \le 1)$.

2.2. Hardware platform

The considered hardware platform is a heterogeneous NoC-based reconfigurable platform, which is implemented on an FPGA platform as shown in Figure 1. The platform consists of a set of different processing elements (PEs) that are connected by a communication NoC. The NoC architecture uses 2D-Mesh topology, packet switching, wormhole combined with virtual channel

flow control, and XY routing algorithm. Each PE can support either hardware or software tasks. Software tasks are executed in instruction set processors (e.g., Microblaze or ARM), while hardware tasks are executed in reconfigurable regions or intellectual property cores (IPs). In this work, the ISP is responsible for different system management tasks including task mapping, task scheduling, resource control, and reconfiguration control and can support one or more tasks.

Definition 2: A NoC-based reconfigurable platform is denoted by a directed graph $NoC = N(P, R, L)$ where P is the set of processing elements, R is the set of routers, and L is the set of physical connections connecting between any two routers.

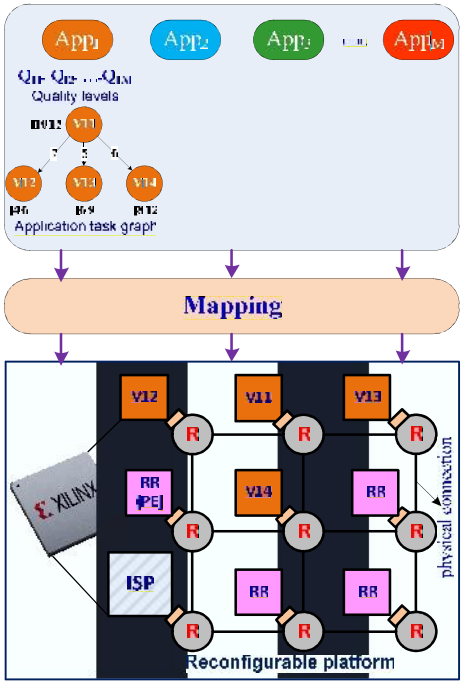

Fig. 1. System model

A PE $p_{xy} \in P$ is represented by attributes ($p_{id}, p_{add}, p_{type}, p_{use}$), where p_{id} is the PE identifier, p_{add} is the PE address used to receive packets, p_{type} is the PE type (p_{HW}: hardware or p_{SW}: software) and p_{use} is the status of PE (used or free).

3. Run-time mapping problem

3.1. Problem formulation

We extend the mapping problem in [11] as follows.

Given the ATGs of M incoming applications and the platform status

Find M near convex regions on the platform and the mapping function map() for application App_i into region R_i, $\forall v_{ik} \in V$, $map(v_{ik}) \rightarrow p_{xy}$ *in* $R_i, p_{xy} \in P$ with the objectives:

$$B = \frac{1}{M}\left\{\sum_{i=1}^{M} \ell_i \sum_{j}^{N_i} \alpha_{ij} B_{ij}\right\}_{\max} \quad (1)$$

and

$$T = \left\{\sum_{i=1}^{M}\sum_{j=1}^{N_i} \alpha_{ij} t_{ij}\right\}_{\min} \quad (2)$$

Such that $\sum_{j=1}^{N_i} \alpha_{ij} = 1$, $\alpha_{ij} = 1$ if Q_{ij} is selected, 0 otherwise.

$$\begin{cases} \sum t_{HW} \le \sum p_{HW} \\ t_{comp_{ij}} \le t_{reqcomp_{ij}} \\ t_{comm_{ij}} \le t_{reqcomm_{ij}} \end{cases} \quad (3)$$

3.2. Near convex region selection strategy

Selecting contiguous near convex regions on the platform for mapping applications tasks is an effective solution, which has been proven in studies [14] and [15]. However, there are some limitations that need to be improved and overcome in these studies. The Neighboraware Frontier (NF) region selection strategy (RSS) in [14] tries to find available PEs in the platform to form a convex region for incoming applications with total minimum Manhattan distance. This strategy assumes that the platform has one or several regions, which have already existed. A new region is created by refering to the existing regions. Therefore, the effectiveness of the new region will depend on the regions created earlier. In addition, the NF strategy may select a non-near convex region. For example, when an application with 9 tasks enters the system, the NF strategy may select a 2x5 region instead of selecting a 3x3 one. Considering a different approach, the study in [15] creates a near convex region for the incoming applications by selecting a firstly available PE, then increasing Hop distance to find out the next PEs of the region surrounding the first PE. This approach can find an optimal near convex region. However, it can also generate non-contiguous regions or increase the fragmentation for PEs on the platform when there are many incoming applications, causing disadvantages for the later incoming applications. This may reduce the overall performance of the system.

In this study, we propose a new near convex region selection strategy that aims to provide flexible resource allocation, fast implementation, and reduced average Manhattan distance in the selected region. The strategy is divided into two steps. In Step 1, a number of hard/soft PEs corresponding to the quality levels of the applications are allocated. In Step 2, an optimal near convex region based on the method of geometrical scanning angle and using minimum Manhattan distance constraint is found.

- Allocation of hard/soft PEs

When a running application wants to change its quality level or when a new applicaton enters the system, the Pes should be re-allocated to meet the QoS requirements of the applications. The question is, how many hard/soft PEs should be allocated and according to what criteria? The answer depends not only on the status of available resources, but also on the complexity and priority level of the applications as well as the QoS requirements of the user. If the applications have the same priority, our approach will allocate hard/soft PEs evenly among the applications. If the applications have different priorities or request different quality levels, the number of hard PEs will be prioritized to allocate more for applications with higher priority or request to run at higher quality level.

- Near convex region selection

Once a number of PEs have been allocated for an incoming application, we need to find a region on the platform to map these Pes. For this purpose, we propose a new region selection technique based on geometrical scanning angle combined with the minimum Manhattan distance to select a near convex region for the incoming application. This technique is described as follows: first, a center PE or near center PE on the platform called the scanning center is found. Then, to find all available PEs on the platform, scanning angles ranging from 0 to 360^0 are used. The scanning direction can be clockwise or counterclockwise. At a certain scanning angle, we may find available PEs. By applying other scanning angles, we can find the number of available PEs needed for the incoming application. To make sure to create a near convex region, we use the minimum Manhattan distance to select the available PEs close to each other. Because the scanning angles are continuous, the fragmentation as well as the distances of contiguous regions are reduced to a minimum level. After each scanning, the scanning angle status will be updated to be used for the next scan when a new application joins the system. The pseudo-code of our near convex region selection strategy is presented in Algorithm 1.

Algorithm 1: Near convex region selection

Input: *A(V,E), N_task, N(P,L)* //N_task $v_{ik} \in V$; PE $p_{xy} \in P$

Output: *R(N,L)* // selected region

```
1: BEGIN
2:    R = ∅;
3:    integer sotf_PE = K*N_task; // the number of soft PEs
4:    integer hard_PE = N_task – sotf_PE; // the number of hard PEs
5:    if N_task > resource_available then goto step 28;
6:    else
7:        if remain_PE _angle < N_task then
8:            update_value(S);
9:            sort( S, compare );
10:       end if
11:       if sotf_PE ≠ soft_PE_available then
12:           sotf_PE = soft_PE_available;
13:           hard_PE =N_task - sotf_PE;
14:       else
15:           if hard_PE ≠ hard_PE_available then
16:               sotf_PE = N_task - hard _PE_available;
17:               hard_PE =hard_PE_available;
18:           end if
19:       end if
20:       for mark_PE_angle to max_PE_angle do
21:           insert(PE in S to R);
22:           if enough sotf _PE and hard_PE then break;
23:           end if
24:       end for
25:   update(mark_PE_angle, hard_PE_available, soft_PE_available);
26:   update(remain_PE _angle);
27:   end if
28:   END
```

3.3. Run-time heuristic mapping algorithm

The purpose of the mapping is to put the tasks of the application into the near convex region that has been selected in the previous step so that the quality level of the application is mazimized while the communication latency of he application is kept at a minimum level.

We propose a run-time heuristic mapping algorithm that focuses on optimizing the tasks with large total computation and communication time. The

algorithm is descirbed as follows. First, the algorithm searches for a hard PE with total Manhattan distance to other PEs is smallest in the selected regions, and at the same time the number of available neighbor PEs to it is largest, called the first PE (center). Then it sorts the tasks in ATG in descending order of total communication time (using Quicksort) and sorts neighbor tasks in the descending order of the total communication time. After that, the first task is mapped into the first PE. Next, the neighbor tasks of the first task are mapped into the neighbor PEs of the first PE in descending order of communication time. Then the task with the largest total communication time among the mapped tasks excluding the first task is updated. Next, the algorithm continues to map the remaining neighbor tasks of the already updated tasks to the available PEs so that the total Manhattan distance to the mapped linking tasks is minimal. The process is performed repeatedly until all remaining tasks are mapped. The pseudo-code of the mapping algorithm is shown in Algorithm 2.

Algorithm 2: Heuristic task mapping

Input: *A(V,E), R(P,L)*

Output: *tmpg* (mapping *A(V,E)* → *R(P,L)*)

1: **BEGIN**
2: *pe_center* = ***max_pe_idle_center***(*tile_info*);
3: ***sort***(*task_vector*, decrease_t_comm);
4: *task_cur* = *task_vector.begin*;
5: ***map_task_pe***(*task_cur,pe_center*);
6: *task_vector_child* = ***task_neighbor***(*task_cur,tile_info,app_graph*);
7: ***sort***(*task_vector_child*, decrease_t_comm);
8: **for all** unmapped *taski* ∈ *task_vector_child* **do**
9: *id_pe_map* = ***findPE_1_hop***(*task_cur,tile_info*);
10: **if** *id_pe_map* != -1 **then**
11: ***map_task_pe***(*taski, id_pe_map*);
12: **end if**
13: **end for**
14: **for all** unmapped *taski* ∈ *task_vector_child* **do**
15: *id_pe_map* = ***pe_nearest*** (*taski,tile_info,app_graph*);
16: ***map_task_pe***(*taski,id_pe_map,tile_info*);
17: **end for**
18: *task_next* = ***find_next_task***(*task_vector,tile_info*);
19: **while** *tile_infor_ unallocated*.size != 0 **do**
20: *task_vector_child*= ***task_neighbor***(*task_next,tile_info,app_graph*);
21: ***sort***(*task_vector_child*, decrease_t_comm);
22: **for all** unmapped $task_i$ ∈ *task_vector_child* **do**
23: *id_pe_map* = ***pe_nearest*** ($task_i$,*tile_info,app_graph*);
24: ***map_task_pe***($task_i$,*id_pe_map,tile_info*);
25: **end for**
26: *task_next* = ***find_next_task***(*task_vector,tile_info*);
27: **end while**
28: **END**

4. Simulation results and discussion

4.1. Simulation setup

The simulations are performed on the same platform as described in [11]. The ISP can accomodate up to six tasks while each reconfigurable region can run only one task. Ten synthetic applications are used in our simulations. These applications have different sizes ranging from 5 to 12 tasks and they are generated by TGFF tool in [19]. For simplicity reason, we assume that each application i has four different quality levels (i.e., Q_{i1}, Q_{i2}, Q_{i3} and Q_{i4}). The computation time of tasks, communication time between pair of tasks are generated as described in [11].

Two scenarios have been implemented in the simulations: (i) The applications are deployed on the platform at the same time. (ii) The applications are deployed sequentially on the platform in order of appearance. Each scenario are run 20 times on 5x5 platform with the total number of tasks of running applications varying from 25 to 30.

4.2. Results and discussion

Three heuristic algorithms are chosen to compare with our heuristic: First Fit (FF), Nearest Neighbor (NN) in [18] and Chen in [15]. These algorithms map the tasks of application on the near convex region generated from NF and our region selection strategy. The parameters we use to comprare the four algorithms include Overall Benefit (OB), quality level, Average Manhattan Distance (ADM), Average Communication Manhattan Distance (ACMD), and run-time of the algorithms.

4.2.1. Quality level of applications evaluation

As presented in Section 2, benefit values are used to represent the satisfaction level of user when receiving a certain quality level. When user receives the highest quality of an application, the value of benefit is 1. A lower quality has the benefit smaller than 1. Therefore, we use it to evaluate the achieved quality for applications. Figure 2 shows the value of achieved OB when deploying applications on the platform according to different algorithms. In all cases, our heuristic algorithm gives better OB than

FF, NN and Chen [15] and an average improvement of 43% of the OB can be observed.

In a specific case, we have deployed three applications (App1 - 7 tasks, App2 - 10 tasks and App3 - 11 tasks) simultaneously on the platform. The OB value and the achieved quality level in this case are shown as Table 1. It is easy to see that our mapping algorithm gives the best results in comparison with other algorithms.

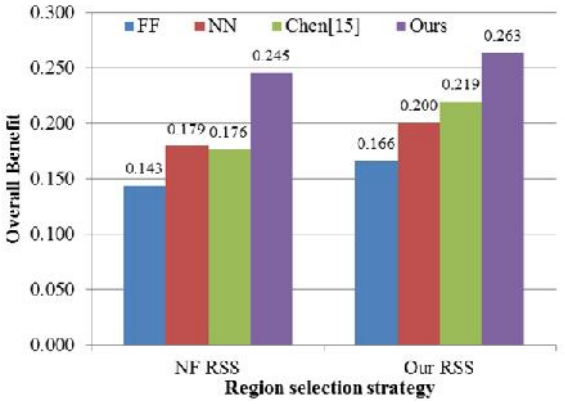

Fig. 2. Overall benefit of mapped applications

Table 1. Quality and benefit values of the applications

Algori-thms	App_i	Quality level		Overall benefit	
		NF RSS	Our RSS	NF RSS	Our RSS
FF	App_1	Q_{14}	Q_{14}	0.080	0.121
	App_2	Q_{24}	Q_{22}		
	App_3	Q_{33}	Q_{33}		
NN	App_1	Q_{14}	Q_{14}	0.142	0.162
	App_2	Q_{22}	Q_{22}		
	App_3	Q_{32}	Q_{31}		
Chen [15]	App_1	Q_{13}	Q_{12}	0.130	0.192
	App_2	Q_{23}	Q_{22}		
	App_3	Q_{32}	Q_{32}		
Ours	App_1	Q_{13}	Q_{11}	0.182	0.290
	App_2	Q_{22}	Q_{22}		
	App_3	Q_{31}	Q_{31}		

4.2.2. Performance evaluation

Next, we evaluate the network performance of the algorithms through two parameters AMD and ACMD. The AMD is the ratio between total Hop of mapped applications over the total edges in task graph of applications. An application mapping algorithm has lower AMD then latency and energy consumption will be smaller because the data packets passes through a smaller number of Hop. The ACMD represents the communication time of data packets through each hop. The ACMD also represents the latency and energy consumption of the network, as it relates directly to the number of data packets and number of Hops each data packet passes through.

The AMD and ACMD values are shown in Table 2. The values of these parameters of our heuristic algorithm are smaller than those of others. This also means that our algorithm has lower latency and energy consumption compared with other algorithms.

Table 2. AMD and ACMD values of algorithms

Algorithms	AMD		ACMD	
	NF RSS	Our RSS	NF RSS	Our RSS
FF	2.023	1.857	2.070	1.874
NN	1.833	1.704	1.828	1.713
Chen[15]	1.943	1.788	1.860	1.691
Ours	1.816	1.594	1.592	1.430

4.2.3. NF vs. Our region selection strategy evaluation

Selecting a near convex region then mapping application into it not only brings performance benefit, defrags for PEs on the platform, but also helps to rapidly deploy applications on the system. In this subsection, the run-time of the algorithms and ratio of OB, AMD, ACMD parameters between our region selection strategy and of NF are evaluated. The run-times of the algorithms are shown as in Table 3 when deploying applications on platforms with the size of 5x5, 6x6, and 7x7. Our strategy has a lower run-time than NF because it's simpler.

Table 3. Run time of region selection algorithms

Platform size	# of task	Average run time (ms)		Gain (%)
		NF RSS	Our RSS	
5x5	25→30	0.1165	0.0704	-39.55
6x6	36→41	0.2129	0.1018	-52.20
7x7	49→54	0.3913	0.1302	-66.72

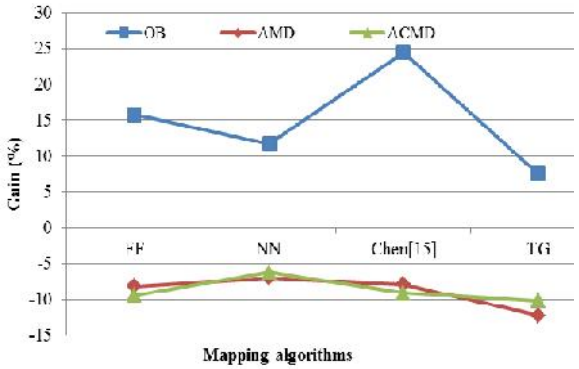

Fig. 3. Improvements of our RS strategy

Next, Figure 3 shows gain obtained by our region selection strategy over the NF in terms of OB, AMD, and ACMD. The OB value in our region selection strategy has an average improvement of

12.2 %. The AMD and ACMD values are decreased by 8.6 % and 8.3 %, respectively. The results show that our region selection strategy is more efficient than NF.

5. Conclusion and future work

In this paper, we have proposed an efficient technique for mapping at run time hardware/software tasks of quality adjustable applications on the reconfigurable platform under limited resources including a near convex region selection strategy and a dynamic heuristic mapping algorithm. The simulation results show that our approach is more flexible, uses resources more efficiently and achieves higher quality of service compared with existing approaches. In the future, we will consider platform with different types of reconfigurable regions to improve performance, flexibility and energy saving for the system.

References

[1] http://www.xilinx.com/products/silicon-devices/soc.html

[2] Kim, Dong-Jin, Yeon-Jeong Ju, and Young-Seak Park, "An Implementation of SoC FPGA-based Real-time Object Recognition and Tracking System," IEMEK Journal of Embedded Systems and Applications, vol. 10, no. 6, pp. 363-372, 2015.

[3] Luo, Junwen, Graeme Coapes, Terrence Mak, Tadashi Yamazaki, Chung Tin, and Patrick Degenaar, "Real-Time Simulation of Passage-of-Time Encoding in Cerebellum Using a Scalable FPGA-Based System," IEEE transactions on biomedical circuits and systems, vol. 10, no. 3, pp. 742-753, 2016.

[4] Flasskamp, Martin, Gregor Sievers, Johannes Ax, Christian Klarhorst, Thorsten Jungeblut, Wayne Kelly, Michael Thies, and Mario Porrmann, "Performance estimation of streaming applications for hierarchical MPSoCs," In Proceedings of the 2016 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, ACM 2016.

[5] Leibo, L. I. U., W. A. N. G. Dong, C. H. E. N. Yingjie, Z. H. U. Min, Y. I. N. Shouyi, and W. E. I. Shaojun, "An Implementation of Multiple-Standard Video Decoder on a Mixed-Grained Reconfigurable Computing Platform." IEICE transactions on Information and Systems 99, no. 5, pp. 1285-1295, 2016.

[6] Hsiao, Pei-Yung, Shih-Yu Lin, and Shih-Shinh Huang, "An FPGA based human detection system with embedded platform," Microelectronic Engineering 138, 2015, pp. 42-46.

[7] Pang, Ke, Virginie Fresse, Suying Yao, and Otavio Alcantara De Lima, "Task mapping and mesh topology exploration for an FPGA-based network on chip," Microprocessors and Microsystems 39, no. 3, pp. 189-199, 2015.

[8] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," Computer (Long. Beach. Calif), vol. 35, no. 1, pp. 70–78, 2002.

[9] R. Kumar, D. M. D. M. Tullsen, P. Ranganathan, N. P. N. P. Jouppi, and K. I. Farkas, "Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance," in Proceedings of the 31st annual international symposium on Computer architecture, 2004, pp. 64–75.

[10] M. S. Abdelfattah, A. Bitar, and V. Betz, "Take the Highway : Design for Embedded NoCs on FPGAs," in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2015, pp. 98–107.

[11] N. Van Cuong, N. T. Bang, L. D. Tuyen, and P. N. Nam, "Dynamic Mapping of Quality Adjustable Applications on NoC-based Reconfigurable Platforms," in The International Conference on Advanced Technologies for Communications (ATC), 2016, pp. 321–326.

[12] L. Ost, G. M. Almeida, M. Mandelli, E. Wachter, S. Varyani, G. Sassatelli, L. S. Indrusiak, M. Robert, and F. Moraes, "Exploring heterogeneous NoC-based MPSoCs: From FPGA to high-level modeling," in 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip, ReCoSoC 2011 - Proceedings, 2011, pp. 1–8.

[13] A. K. Singh, T. Srikanthan, A. Kumar, and W. Jigang, "Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms," J. Syst. Archit., vol. 56, no. 7, pp. 242–255, 2010.

[14] Chen. L. Chou, U. Y. Ogras and R. Marculescu, "Energy-and Performance-Aware Incremental Mapping for Networks on Chip With Multiple Voltage Levels," IEEE Transactions on ComputerAided Design of Integrated Circuits and Systems, vol. 27, no. 10, pp. 1866-1879, 2008.

[15] Chen. L. Chou and R. Marculescu, "User-aware dynamic task allocation in networks-on-chip." 2008 Design, Automation and Test in Europe. IEEE, 2008.

[16] M. Haghbayan, A. Kanduri, A. Rahmani, P. Liljeberg, A. Jantsch, and H. Tenhunen, "MapPro : Proactive Runtime Mapping for Dynamic Workloads by Quantifying Ripple Effect of Applications on Networks-on-Chip," in Proceedings of the 9th International Symposium on Networks-on-Chip, p. 26. ACM, 2015.

[17] N. P. Ngoc, G. Lafruit, S. Vernalde, and R. Lauwereins, "Real-Time 3D Applications on Mobile Platforms With Run-Time Reconfigurable region Accelerator," In Proceedings of the International Conference on Computer Visions and Graphics, ICCVG, 2002, pp. 582-588.

[18] E. Carvalho, N. Calazans, and F. Moraes, "Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs," in 18th IEEE/IFIP International Workshop on Rapid System Prototyping, 2007. RSP 2007, 2007, pp. 34–40.

[19] R. P. Dick, D. L. Rhodes and W. Wolf, "TGFF: task graphs for free," Proc. Intl.