Task mapping and mesh topology exploration for an FPGA-based network on chip

Ke Pang, Virginie Fresse, Suying Yao, Otavio Alcantara de Lima

Task mapping and mesh topology exploration for an FPGA-based network on chip

Ke Pang[12], Virginie Fresse[2], Suying Yao[1], Otavio Alcantara De Lima Junior[2]*

[1]School of Electronic and Information Engineering, Tianjin University, 92 Weijin Road, 300072 Tianjin, China
[2]Laboratoire Hubert Curien, UMR CNRS 5516 18 rue Benoit Lauras, Jean Monnet University, University de Lyon, 42000 Saint Etienne, France

ABSTRACT

Task mapping strategies on NoC (Network-on-Chip) have a huge impact on the timing performance and power consumption. So does the topology. In this paper, we describe the exploration flow of task mapping algorithms using different NoC mesh shapes. The flow is used to evaluate timing and energy consumption based on a NoC emulation platform. It is open to any task mapping algorithms and to any shapes of NoC mesh. A heterogeneous (PC and FPGA) platform is used to fully perform each step of the flow. The experiments demonstrate that the most appropriate task mapping strategy and the most suitable NoC shape strongly depend on the algorithm used. Depending on the timing latency results obtained and the FPGA resources used, the designer can select the appropriate task mapping strategy on the suitable shape in a short exploration time and with precise timing evaluation.

Keywords Task mapping exploration flow, Task mapping strategy, NoC shape, NoC emulation platform, FPGA resources, Timing latency

1. INTRODUCTION

With the mature technology of modern integrated circuit process, System-on-Chip (SoC) with multi-core has undergone great advances. SoC can incorporate numerous Intellectual Property (IP) cores that perform different functions, possibly at different clock frequencies. With the increasing number of IPs on SoC, the design of underlying communication architecture has a major impact on the performance and energy consumption of the overall system.

As a promising alternative to the traditional bus and point-to-point connection, Network-on-Chip (NoC) is a design paradigm and on-chip architecture. NoC can connect all the IP cores to a router-based network using an appropriate Network Interface. This architecture greatly helps overcome the design problems of current bus-based SoC methodologies.

With the development of FPGA technology, hardware-based emulation using the evaluation platform has become widely used. Compared with software simulation, it can accelerate the validation process as it significantly reduces the system evaluation time [9]. With shortening of the cycle of the NoC design, development costs will be reduced.

Fig. 1 depicts the traditional application-specific NoC design flow [1] [2]. One application can be described as a task graph. The task graph is mapped on one NoC using one task mapping algorithm. The mapping solution must fulfill specific requirements (e.g. reduce energy consumption and congestion) as well as reduce the total communication delay. Similar to an optimal mapping strategy, the appropriate NoC architecture parameters can reduce the resources used and further improve performance. In the generic NoC design flow, the

*Corresponding author.
Email address: cocopang324@hotmail.com (K Pang); virginie.fresse@univ-st-etienne.fr (V. Fresse); syyao@tju.edu.cn (S. Yao); otavio.alcantara.de.lima.junior@univ-st-etienne.fr (O. Alcantara De Lima Junior)

topology and the number of routers are usually defined in the NoC specification. The topology mostly used is a regular mesh with the same number in the X and Y axis. Such design flow does not explore the topology or the shape of the NoC and does not consider several task mapping algorithms. The jointly exploration of both the shape of the NoC and the mapping solutions has not been yet proposed. The contributions of this paper are:

- The analysis of the shapes of the NoC and the impact on the performances according to the number of tasks in the application.
- The exploration flow developed to jointly explore the shape of the NoC and the task mapping algorithms.
- The analysis of the results (timing on FPGA, energy consumption and the mapping time) according to the NoC specifications and the application.

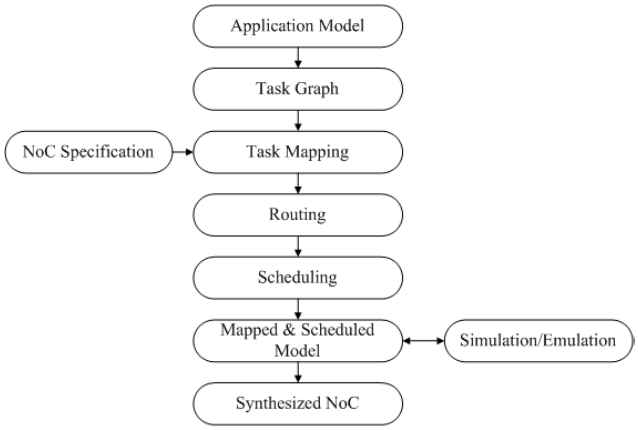


Fig. 1. Traditional application specific NoC design flow

In this paper, we propose one design flow to jointly explore task mapping algorithms and mesh topology on NoC. This exploration flow can evaluate timing and dynamic energy consumption based on the NoC emulation platform. The platform used is a heterogeneous (PC and FPGA) platform that fully performs the exploration flow. Timing evaluation is performed on the dedicated NoC emulation platform on FPGA to fully explore all task mapping and mesh topology solutions.

This paper is organized in 6 sections. Section 2 presents the related works on task mapping techniques and NoC emulation platforms and discusses the corresponding problems. Section 3 explains the exploration flow in detail. Section 4 is dedicated to the experimental study of the exploration flow. Section 5 discusses the implications of the experimental results in Section 4. Finally Section 6 presents the conclusions and the future outlook for exploration flow.

2. RELATED WORK

In recent years, several studies have been conducted on mapping the tasks of the application on NoC architecture [3] [4] [5] [6] [7]. Their task mapping algorithms use either dynamic or static mapping. Run-time incremental mapping [3] and congestion aware algorithms [4] are the most widely used dynamic mapping techniques. They are performed during the execution of the application. They can regulate the mapping strategy online according to the systemic feedback. The two-step genetic algorithm (GA) [5], the Branch-and-Bound Algorithm (BB) [6] and the template-based efficient mapping algorithm (TEM) [7] are typical static mapping techniques. These techniques are implemented offline before the application runs. Static mapping solutions are generally recommended as the computational overhead of dynamic mapping algorithms increases the overall delay and the energy consumption of the system [2].

Some authors have reviewed the different NoC task mapping techniques. In [2], the authors review most task mapping strategies. They compare studies and give the communication cost and the execution time using a set of benchmarks (VOPD, MPEG 4, PIP and task graphs generated by the TGFF tool [18]). The communication cost corresponds to the absolute communication cost (hops × bandwidth). The CPU time required to generate the task mapping solution is also given for each algorithm. Experiments are based on the 8×8 mesh NoC for 64-core applications and the 8 × 16 mesh NoC for 128 core applications. In [8], an evaluation of dynamic mapping and static mapping based on MPSoC architecture is described. In

the simulations of single application mapping and multiple application mapping, the latency, congestion and energy consumption performances of the task mapping algorithms are given.

In all the above papers, the results of these techniques are only simulated. The results are based on NoCs using the regular N×N mesh topology. No studies explore the shape of the NoC (shape meaning the X and Y number of routers in the mesh topology).

Hardware-based emulation using the evaluation platform is widely used to accelerate the validation process [9]. Emulation based on FPGA technology provides a rational way for the NoC. Many emulation platforms are designed for exploration of the NoC [10] [11] [12] [13]. Most emulation platforms aim to explore NoC topology or NoC interconnections. The exploration selects the most appropriate topology from several available topologies, but does not consider task mapping. XNoC [14] is an emulation platform for the evaluation of the mapping algorithm. This evaluation environment can support both static and dynamic application mapping for performance evaluation and cost metrics. But this tool cannot adequately evaluate and compare different task mapping techniques directly.

We propose an exploration flow based on the NoC emulation platform for the NoC designer. We focus on evaluating different task mapping techniques with different shaped NoC router arrays for the NoC implementation of the specific application. In this way, the most appropriate task mapping solution and the FPGA resources used for NoC design can be obtained.

3. EXPLORATION FLOW

Fig. 2 shows the system architecture of the task mapping exploration flow. It combines NoC configuration parameters and application communication parameters with the task mapping algorithms. It explores the most appropriate task mapping algorithm combined with the most suitable shape of the NoC for one specific application.

The exploration flow is implemented on a PC and FPGA-based NoC emulation platform. Many existing task mapping algorithms are described in C/C++ so that task mapping scenarios can be conveniently generated with PC. The FPGA-based emulation platform can accelerate the emulation process and provides the real-time timing results for analytical purposes.

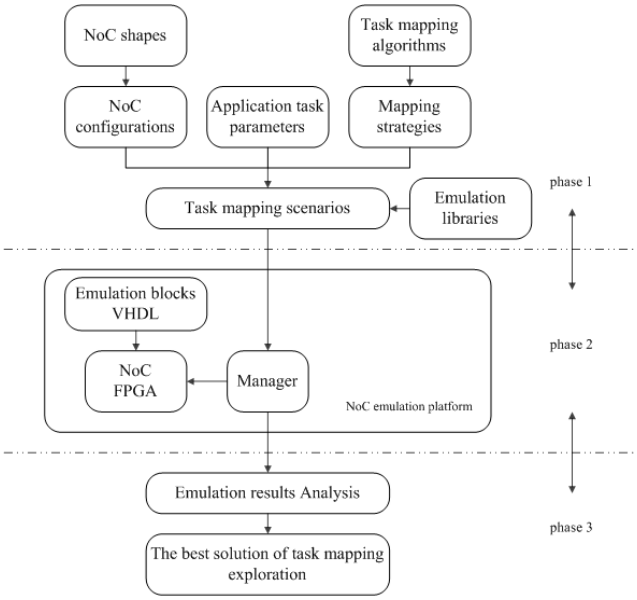


Fig. 2. The system architecture of the task mapping exploration flow

The inputs of the exploration flow are the following: the NoC configuration parameters (several shapes of the NoC), the communication parameters of the application, and several task mapping algorithms for the NoC. The system generates the task mapping scenarios using all the input parameters and the mapping strategies generated by the task mapping algorithms. These task scenarios configure the NoC emulation platform to implement the application on the FPGA. The FPGA-based emulation platform emulates the application and gives the emulating results. In this case, the result is the timing latency needed to complete the data transmission between the routers.

The output of the exploration flow is the best task mapping solution for the specific application on the specific NoC shape.

Details of the inputs and output of the flow are discussed in the following subsections.

3.1 NoC configurations

The NoC configuration parameters are used to configure NoC to generate the required NoC communication architecture.

Here, the NoC architecture is designed with the mesh topology, which is the most suitable topology for FPGA implementation. Based on the mesh topology, the size of the NoC is defined as X×Y: the router number of each line – X and the router number of each column – Y. X×Y represents the shape of NoC structure used in the generation of the task mapping scenario. Fig. 3 shows an example with 4×4 NoC architecture. The only constraint linked to the NoC size configuration is the total number of routers – X×Y – has to be higher than the number of tasks in the specific application required to map the application on NoC. Here we specify several shapes as a function of the structure of the task graph concerned.

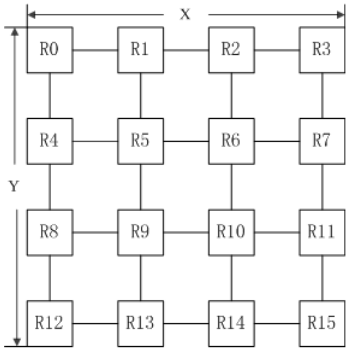
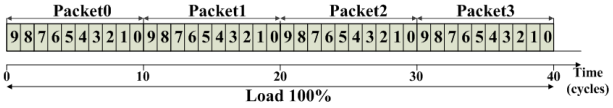


Fig. 3. Example of 4x4 NoC architecture

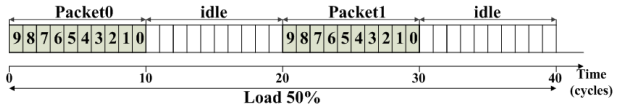
3.2 Task Graph Configuration

One application can be described as one set of tasks, called a task graph. Each task graph defines the relationship between the task nodes and gives the parameters needed to describe the communication [10] between the task nodes. The communication parameters are defined according to the requirements of the FPGA-based NoC emulation platform [20] and the parameter values depend on the requirements of the specific application.

Based on the emulation platform, data transmission from the source router to the destination router is imitated by the message. A message consists of a set of packets (defined by the number of packets), and a packet is a set of Flits (Flow control Units) (defined by the size of the packet). Here, the size of flit is defined as 16 bits. Packets are sent according to the data injection charge. The data injection charge is defined as the idle time between two packets of one message. It indicates the ratio of the bandwidth to the packets used. In Fig. 4, the data injection charge (or load) is 100% for the first case (a), and 50% for the second case (b).



(a) Load 100% (injCharge = 0)



(b) Load 50% (injCharge = 10)

Fig. 4. The examples of data injection charge

Usually one message is transferred in several batches [20] (the number of batches is defined by the designer and here we set the number of batches as 10). Each batch consists of several packets. This is to avoid one message occupying the transmission channel for too long.

The application task parameter file provides all the parameters required for communicating between task nodes. These communication parameters [10] are used to configure the transmission of the data on the NoC emulation platform. One task graph file usually includes the three sections shown in Table 1.

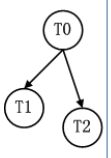
Table 1 Three sections of the application task graph file

Keyword	Definition	Type	Value1	Value2
TASK	task number	task type	(reserved)	(reserved)
ARC	arc number	Arc type	SRC_TASK	DEST_TASK
PARA	parameter name	type number	parameter value	(reserved)

For each application task graph, there are three sections in the file, as depicted in Fig. 5. The ‘TASK’ section lists all the tasks with their task

types. The ‘ARC’ section describes the connections between task nodes with their arc types. The ‘PARA’ section gives the tables of the required five parameters corresponding to the task or arc type:

1) The size of each packet of each transmission between two tasks corresponds to the arc type (‘packetSize’). For example, in Fig. 5 from TASK0 to TASK1, the size of the packet is 21 flits;
2) The number of packets to be sent between two tasks, corresponds to the arc type (‘packetNum’). For example, from TASK0 to TASK1, the number of packets is 36;
3) The data injection charge between two flits of every transmission between two tasks, corresponds to the arc type (‘injCharge’). For example, from TASK0 to TASK1, the injection charge is 76 clock cycles;
4) The period between two batches, corresponding to the task type (‘period’). For example, from TASK0 to TASK1, the period is 1171 clock cycles;
5) The latency constraint (hop number) between two tasks, corresponds to the arc type (‘latency’). The latency constraint for NoC mapping represents the number of hops between two communication routers corresponding to two tasks. It shows the maximum distance between two task nodes when they are mapped. For example, from TASK0 to TASK1, the latency constraint is 4 hops.




```
TASK    t0      TYPE 18
TASK    t1      TYPE 15
TASK    t2      TYPE 4

ARC a0      FROM t0 TO t1   TYPE 37
ARC a1      FROM t0 TO t2   TYPE 28

PARA  packetSize        PARA  packetNum       PARA  injCharge
Type  value             Type  value           Type  value
28     18               28    19              28    59
37    21                37    36              37    76

PARA  period            PARA  latency
Type  value             Type  value
 18   1171              28    3
 15   1264              37    4
  4   1138
```

Fig. 5. One example of an application task graph

All the communication parameters of the application task graph in the file will be extracted by the system as inputs for the generation of the task scenario.

3.3 Task mapping algorithm

There are many task mapping algorithms for NoC as mentioned in Section 2. The exploration flow is open to all static mapping algorithms. Among static mapping algorithms, the deterministic mapping algorithms [6] [7] [16] [17] do not depend to a too great an extent on the setting of convergence condition (like GA [5]) or the experience of the designer (like ILP [15]). It is highly optimized for general use to validate the exploration flow.

The Branch and Bound (BB) mapping algorithm [6] is a typical deterministic mapping algorithm. It is a systematic search algorithm that topologically finds the optimal mapping by searching for the solution in tree branches and bounding unallowable solutions. The traditional BB algorithm for NoC takes the minimum possible amount of energy consumption as the bounding and traversals all possible mapping patterns. It leads to the best solution at the cost of CPU time and memory depth. Based on the traditional BB algorithm, M. Reshadi [16] added the bandwidth constraint to intensify the bounds to shorten CPU time.

The template-based efficient mapping (TEM) algorithm [7] takes the constraints of bandwidth and latency into account. Depending on the structure of the task graph, it takes advantage of two templates to generate the mapping solution in only one turn. Compared with the BB algorithm, TEM obtains the mapping solution faster but the quality is not as high.

The bandwidth-constraint and latency-constraint branch-and-bound algorithm (BBL) was developed for the validation of the exploration flow. This algorithm is based on the original branch-and-bound algorithm considering the latency constraint between two tasks [17].

The latency constraint is defined as the router hop numbers between two communication routers that correspond to two task nodes. It indicates the maximum distance allowed between two task nodes when they are mapped. In the same way as

the traditional BB algorithm, the BBL algorithm is executed by alternating two steps: Branch and Bound. Starting from the root node, Branch traverses all the routing paths to map each unmapped node onto the unoccupied router of the array, one by one. A new possible routing path is allocated to generate a new possible mapping pattern and its energy consumption is calculated so it can be compared with the minimum energy cost. The new possible routing path must be deadlock-free. The Bound step uses three bounds to estimate every search path and abort the impossible paths without further trials. These three bounds include the bandwidth requirement, the latency constraint, and energy consumption minimization. Once the search violates the constraints, the attempt to explore this data path is immediately terminated. These bounds trim away the unpromising mapping patterns early in the search and hence speed up the mapping process with less computation time and less memory depth. This new algorithm was developed for this work and is easily inserted in the design flow. Like the traditional BB algorithm, the BBL algorithm is more suitable for small problems on small FPGAs, for example, the image processing application. This is because the implementation time of the algorithm increases considerably with an increase in the size of the NoC. For mapping N tasks to N tiles, the original timing complexity is O(N!). Although the bounds help shorten the implementation, when the NoC is bigger than for example, 7×7 NoC, the mapping solution requires several hours. The details of this algorithm are beyond the scope of this paper.

As already mentioned, the three mapping algorithms are selected with respect to the actual applications or the desired performances. All these algorithms are used to generate the corresponding task mapping strategies and accordingly, to develop the task mapping scenarios with the emulation libraries.

3.4 NoC emulation platform

In this paper, the exploration flow uses one FPGA-based NoC emulation platform with a 7×7 NoC to evaluate the resulting task mapping scenarios. The FPGA-based emulation platform [20] is designed using the SNMP protocol concepts to explore the design space as well as to evaluate the performance of any NoC. This platform enable an easy configuration of emulation blocks and defines an interoperability model based on the MIB description.

The emulation platform connects the emulation blocks (traffic generator and traffic receiver [10]) with the existing Hermes NoC to allow the designer enough space to evaluate different NoC shapes. The Hermes NoC [19] is a packet-switching-based NoC architecture designed by the *Pontifical University Catolica do Rio Grande do Sul*, Brazil. The emulation blocks are designed by the Hubert Curien Laboratory, France. The emulation blocks generate traffic for one directed task graph to trace the communication between the task nodes that are mapped on NoC. They are the specific blocks to insert packets of (Traffic Generators) into and to extract packets of (Traffic Receptors) and from the network [10]. The number of traffic generators (TGs) and traffic receptors (TRs) depends on the size of the NoC. These emulation blocks are managed by the manager. The manager [20] is an agent between the PC and the emulation platform that is implemented in both hardware and software components. Firstly, it transfers the information of task mapping scenarios to the emulation blocks. Secondly, the manager decodes and executes the commands to guide and monitor the behavior of the emulation blocks. Finally, the manager extracts and deals with the emulation results from the emulation blocks and then transfers them to the PC. It should be noted that this emulation platform only emulates the communication between routers of NoC. The processing elements (PEs) and the memories are not connected with the NoC emulation platform.

This platform is a heterogeneous (PC and FPGA) platform that can fully perform the exploration flow. It can retrieve the performance of the communication tasks before implementing the real application code.

Based on the emulation platform, the generated task mapping scenarios are used as the inputs and the timing latency of data transmission be-

tween any two routers can be emulated as the outputs. The timing latency between two routers is expressed as the effective amount of clock cycles needed to complete all data transmission from one router to another. The clock cycles of the idle time between two transmissions are not excluded. The timing latency reflects the actual time required for data transmission according to the task graph parameters. Several latency measures are one reference of the timing performance.

The objective of the task mapping exploration described in the paper is to evaluate task mapping algorithms for NoC and NoC shapes. It is open to any emulation methods.

3. 5 Analysis of results

The last stage of the exploration flow is performance evaluation. The timing results from the emulation platform, the FPGA resources used during the emulation, and the total execution time of task scenarios are analyzed.

For one benchmark, various indexes of timing evaluation are analyzed, for example, the total latency of all data transmission, the average latency of the data path, the latency of the longest data path, and the standard deviation of all the data paths. The unit of measure for latency is the number of clock cycles.

At the same time, the FPGA resources used to implement the application are given according to different task mapping scenarios to evaluate the FPGA resources consumption.

The total execution time of the task scenarios consists of two parts: the implementation time of task mapping algorithm on PC, and the configuration and running time on the FPGA emulation platform.

All the results are assessed according to the different task mapping algorithms and the different NoC sizes, respectively.

For different applications, different design requirements have to be met. The task mapping exploration flow provides the performance analysis for different mapping algorithms and NoC shapes, such as transmission efficiency, dynamic energy consumption, FPGA resources use, and execution time. By comparing these analyzed results, the designer can choose the most appropriate task mapping algorithm and the most suitable shape of NoC to meet the specific requirements of a particular application.

In the following section, we detail the steps of the exploration flow. Three task mapping algorithms and four kinds of NoC size were selected to illustrate the flow. Although the example used is a homogeneous architecture, it also works for heterogeneous architecture.

4. EXPERIMENTS

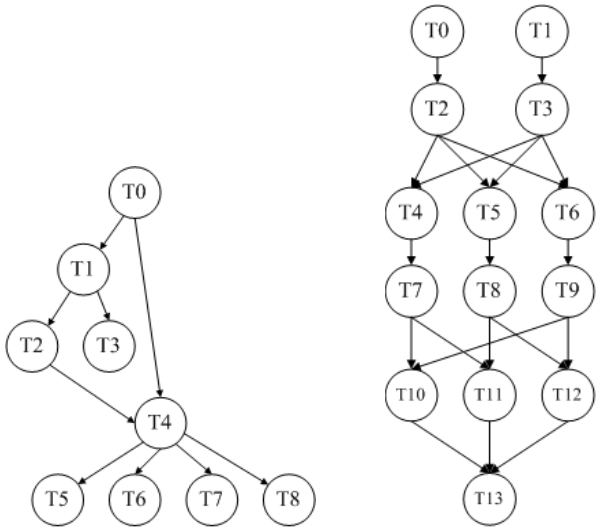
Several task mapping algorithms and mesh topology shapes were selected for the experiments.

For the applications, one random benchmark generated by TGFF [18] (a) and one benchmark of real image applications (multispectral imaging (b)) are selected. The only constraint for the applications is the maximum number of tasks, which needs to be less than 14 to fit to the shapes of the NoC. These two benchmarks are selected to check the validity and feasibility of the task mapping exploration flow.

Fig. 6(a) shows the task graphs of application randomly generated by the TGFF tool. This task graph is a directed acyclic task graph. The maximum number of inputs or outputs of one task node is four. The values of all the necessary communication parameters concerning the transmission between task nodes are randomly generated by the TGFF. Fig. 6(b) shows the task graph extracted from a real image processing application – multispectral imaging. The values of the communication parameters between task nodes are given according to the requirements of the real data transmission. This image application was selected because it has distinct structural features in the connecting relations between task nodes. These two task graphs are implemented on different NoC shapes to check the effects of NoC shape.

The three task algorithms selected are the bandwidth constraint Branch-and-Bound algorithm (BB) [16], bandwidth- constraint and latency constraint Branch-and-Bound algorithm (BBL) and Template-based Efficient Mapping algorithm (TEM) [7]. The BB and TEM are the typical de-

terministic task mapping algorithms. The BBL algorithm was specially designed for this work.



(a) Random benchmark (b) multispectral image

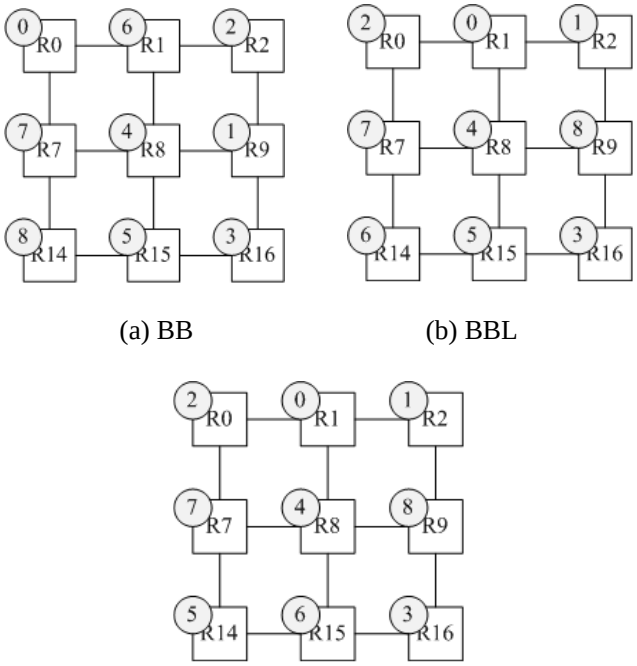
Fig. 6. Directed task graph of a random benchmark and a real application

For the NoC specifications, four shapes of the mesh topology are used on which the application is implemented: 3×3 (only for random benchmark), 4×4, 3×5 and 2×7 respectively. The XY routing algorithm is adopted.

The task mapping scenarios for these benchmarks are generated using PC. The scenarios are then emulated on the ML605 FPGA platform (with the Virtex 6 FPGA). The advantages of such a platform are not only providing logic resources to emulate real communications but also to carry out the performance evaluation in a short time. The main clock frequency of the platform is 66MHz. Its maximum bandwidth can reach 62.94MByte/s.

4.1 The impact of using several algorithms on the 3×3 NoC for a random benchmark

Experiment #1 is focused on the random benchmark in Fig. 6(a). One task node of the benchmark is mapped onto one router of NoC. This task graph has 9 task nodes so NoC shape 3×3 is used in this experiment. With three mapping algorithms, three mapping strategies can be obtained, as shown in Fig. 7.



(a) BB (b) BBL

(c) TEM

Fig. 7. The mapping strategies of the random benchmark on a 3×3 NoC

The squares denoted by 'Rx' in Fig. 7 are the NoC routers and all the circles with the numbers represent the task nodes of the application. The figure depicts the one-to-one relationship between the task node and the router. For example, with the BB algorithm, task node 'T4' is assigned to the router 'R8'.

According to the task mapping solutions for the 3×3 NoC, the total latency time of data transmission, the longest data path latency, and the time needed to implement the task mapping algorithms obtained are listed in Table 2.

Table 2. The results of the experiment of the random benchmark on the 3×3 NoC

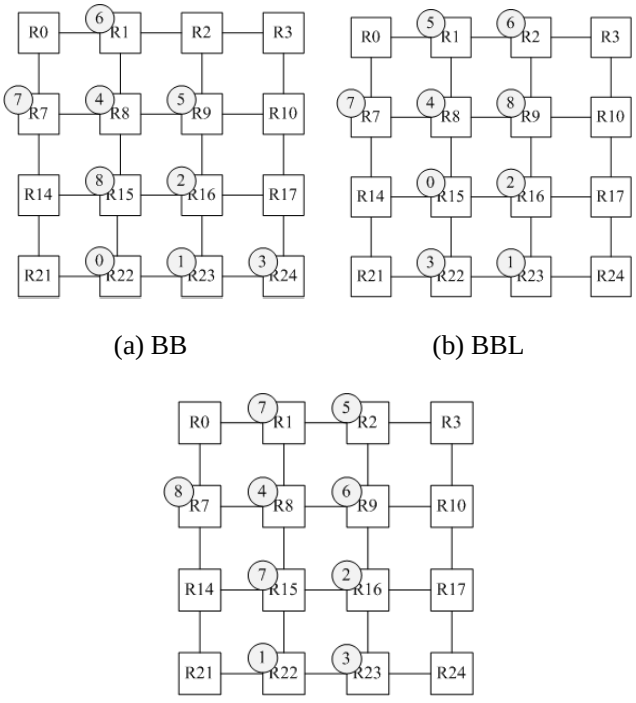
	BB	BBL	TEM
Total latency (clock cycles)	**38954**	43864	42251
Longest path latency (clock cycles)	**87935**	93609	92569
Mapping time (ms)	911808	227952	**4749**

From the point of the shortest value of total data transmission and the longest data path transmission, the best solution for random benchmark

mapped on 3×3 NoC is the BB algorithm. When the implementation time (mapping time) of the algorithm is taken into consideration, the TEM algorithm is the best solution.

4.2 Oversize mapping of random benchmarks on a 4×4 NoC.

Experiment #2 is still focused on the random benchmark, but this time on mapping it on the oversize 4×4 NoC using different task mapping algorithms.

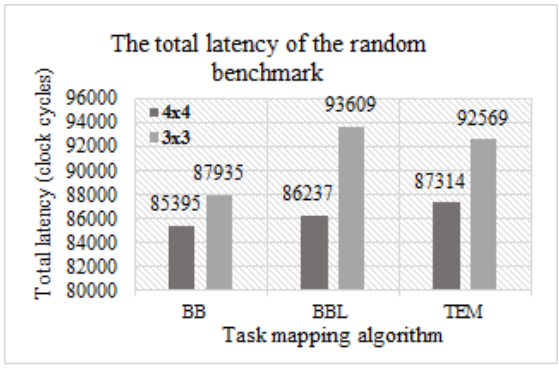


(a) BB (b) BBL

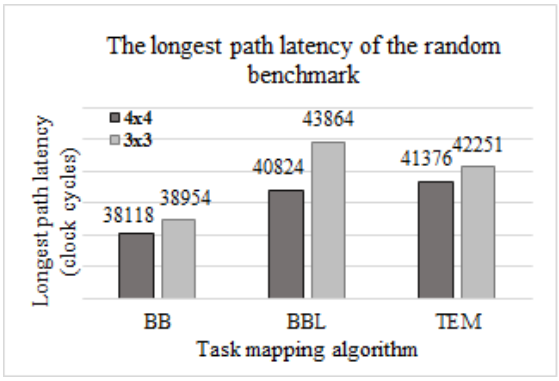
(c) TEM

Fig. 8. Mapping strategies for the random benchmark on the 4×4 NoC

The results of the total latency and the longest path latency of data transmission with different algorithms on the 4×4 NoC and the 3×3 NoC are shown in Fig. 9.



(a)



(b)

Fig. 9. Total latency and the longest path latency of the random benchmark on the 4×4 and the 3×3 NoC with several task mapping algorithms

Diagram (a) in Fig. 9 shows the total data transmission latency. Diagram (b) shows the longest data path latency of data transmission. As can be seen, the total latency of data transmission on the 4×4 NoC can be considerable lower than on the 3×3 NoC. The largest proportion of the decrease can reach 9% with the BBL algorithm. In contrast, the timing latency of the longest data path is only slightly reduced when the 4×4 NoC is used. The trends of the total latency and the longest latency are completely different.

The resources used on the FPGA consist of the number of slices of registers and LUTs listed in Table 3. For example, TEM (3×4) means that the mapping solution is implemented with TEM algorithm on a 4×4 NoC, but the actual NoC used is 3×4. Thereby the resources used (actual resources used compared with configured resources) de-

crease by about 28%. Compared with the strategy for the 3×3 NoC (REG: 1375; LUT: 4640), the ratio of increased resources used is about 28%.

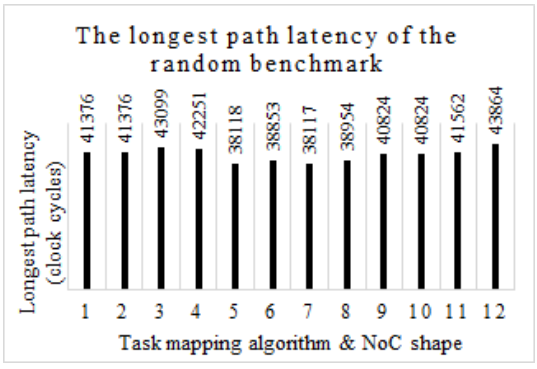
Table 3. The resources used by the random benchmark on 4×4 NoC

	Resources configured		Resources used		Saving ratio (%)		Increasing ratio (%)	
	REG	LUT	REG	LUT	REG	LUT	REG	LUT
BB (4×4)	2660	9176	2660	9176	0%	0%	48%	49%
BBL (3×4)	2660	9176	2660	9176	0%	0%	48%	49%
TEM (3×4)	2660	9176	1914	6537	28%	29%	28%	29%

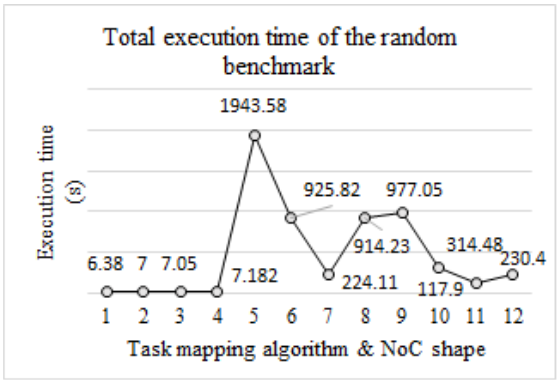
As can be seen by the above analysis, the task mapping solutions with 4×4 NoC have better timing for this random benchmark. However, the 3×3 NoC uses only half the FPGA resources with any task mapping algorithms.

4.3 General mapping experiments with a random benchmark

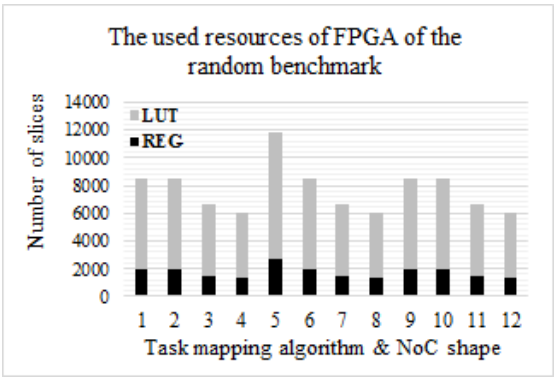
Experiment #3 is still focused on the random benchmark but maps it on different NoC shapes with different task mapping algorithms. The results of the longest path latency of data transmission, the execution time and the resources used with different algorithms on different shaped NoCs are shown in Fig. 10.



(a)



(b)



(c)

1-TEM4×4(3×4) 2-TEM3×5(3×4) 3-TEM2×7(2×5) 4-TEM3×3(3×3)
5-BB4×4(4×4) 6-BB3×5(3×4) 7-BB2×7(2×5) 8-BB3×3(3×3)
9-BBL4×4(3×4) 10-BBL3×5(3×4) 11-BBL2×7(2×5) 12-BBL3×3(3×3)

Fig. 10. The longest path latency, execution time, and resources used for the random benchmark

Diagram (a) in Fig. 10 shows the latency of the longest data path with different algorithms on different shapes. Diagram (b) shows the execution time of the task scenarios. Diagram (c) shows the number of slices of registers and LUTs used on the FPGA according to the different algorithms on different shapes. For example, TEM4×4 (3×4) means that the mapping solution is implemented with the TEM algorithm on 4×4 NoC, but the actual router array used is 3×4.

Table 4 lists dynamic energy consumption and total latency of all task mapping solutions for the random benchmark. From this table, we found that the trend of timing latency is not the same as that of dynamic energy consumption. The designer cannot deduce the timing performance of map-

ping solutions just from the estimation of the dynamic energy consumption, and vice versa.

Table 4. Estimated dynamic energy consumption and the total latency of all task mapping solutions for the random benchmark

		Total latency (clock cycles)	Dynamic energy
BB	4x4	**85395**	**3101**
	3x5	86139	3231
	2x7	87139	3335
	3x3	87935	3465
BBL	4x4	86237	3204
	3x5	86237	3204
	2x7	88230	3646
	3x3	93609	4273
TEM	4x4	87314	3965
	3x5	87314	3965
	2x7	91268	4579
	3x3	92569	4343

The experiments show that each of the different mapping algorithms used for the different shaped NoCs has its own advantages and disadvantages depending on the case concerned. From the different requirement points, different solutions for the random benchmarks can be chosen based on the different requirements. In the sight of the dynamic energy consumption and the total latency, the BB algorithm with 4×4 is the optimal solution. Considering the longest data path latency, the BB algorithm with 2×7 is the best solution; but considering the execution time, the TEM algorithm with 4×4 is the best solution; whereas the 3×3 NoC uses the least FPGA resources for any task mapping algorithm. Therefore, the appropriate mapping algorithm with the appropriate NoC shape needs to be carefully investigated.

4.4 Experiment with an image processing application

In this experiment, the benchmark from the real image processing application – multispectral imaging – is implemented and evaluated. It is evaluated on the FPGA platform with three task mapping algorithms on three shapes: 4×4, 3×5 and 2×7.

First, the total dynamic energy consumption of data transmission is estimated with formula (1). The dynamic energy consumption of one specific application is defined as the sum of the dynamic energy consumption (DE) of all its data paths. It is just the relative statistics for different strategies.

$$DE = \sum_{dataPaths} (packetSize(d) * packetNum(d) * hopNum(d)) \quad (1)$$

Based on the platform, the data from one router to the destination router is transmitted through the message. One message is a set of packets (defined by the number of packets (packetNum)) and one packet is a set of flits (Flow control bits) (defined by the size of the packet (packetSize)) [10]. The dynamic energy consumed on one data path is defined as the product of the amounts of data transmission and the number of hops (hopNum) between the topology nodes of the data path. The amount of data transmission is the number of packets transferred on the data path multiplied by the size of the packet.

Table 5 lists the estimated dynamic energy consumption of each task mapping solution. The solutions using the BB and BBL algorithms on the 3×5 shape use the least dynamic energy.

Table 5. Estimated dynamic energy consumption of all task mapping solutions for multispectral imaging

	4x4	3x5	2x7
BB	5873212	**5873089**	5873212
BBL	5873156	**5873089**	6083033
TEM	6083156	6292743	5873436

At the same time, the timing latency of each task mapping solution with different algorithms on different shapes is emulated on the platform. Table 6 lists the total latency and the longest path latency of all the task mapping solutions of the multispectral imaging application.

Table 6. Timing latency of all task mapping solutions for multispectral imaging

		Total latency (clock cycles)	Longest path latency (clock cycles)
BB	4x4	12178847300	2673824330
	3x5	12110906458	2606284822
	2x7	**5863850013**	3200143801
BBL	4x4	12178685731	**2673789617**
	3x5	12178487399	2673870982
	2x7	5864067418	3200181877
TEM	4x4	5910408782	3433031213
	3x5	8693999935	3394092353
	2x7	9214319094	3625959437

When the timing results in Table 6 are compared with the estimated energy in Table 5, it can be seen that the tendency of timing results is not as same as that of the estimated dynamic energy consumption. Therefore, the designer cannot determine the appropriate task mapping strategy only depending on estimated dynamic energy consumption.

Table 7. Execution time of all task mapping solutions for multispectral imaging

		Mapping time on PC (ms)	Running time on FPGA (ms)	Execution time (ms)
BB	4x4	5435424	5974	5441398
	3x5	3830401	5990	3836391
	2x7	1034352	5974	1040326
BBL	4x4	5435359	5974	5441333
	3x5	3830297	5990	3836287
	2x7	1034332	5974	1040306
TEM	4x4	4789	**5943**	10732
	3x5	4433	6021	10454
	2x7	**4194**	5990	**10184**

Table 7 lists the execution time of all the solutions. It includes the task mapping time on PC and the configuration and running time on the FPGA platform. The mapping time of the TEM algorithm is shorter than that of the other two algorithms, but the running times of three algorithms are similar.

By analyzing the results of the exploration flow based on the emulation platform, the best solutions for this application for different cases can be easily selected according to the different requirements.

5. DISCUSSION

Based on all experimental results of random benchmark and multispectral imaging applications, the best task mapping solutions can be summarized as a function of the different requirements.

Table 8 gives the appropriate task mapping algorithm with the suitable NoC shape for two application benchmarks. The standard used to choose the best solution will differ depending on the case concerned.

Table 8. The best solutions according to different requirements

	Random benchmark		Multispectral imaging	
	algorithm	shape	algorithm	shape
Dynamic energy	BB	4×4	BB/BBL	3×5
Longest path latency	BB	4×4	BB	3×5
Total latency	BB	4×4	BB	2×7
Average latency	BB	4×4	BB	2×7
STDEV of latency	BB	3×3	BBL	2×7
Execution time	TEM	4×4	TEM	2×7
FPGA resources	BB/BBL/TEM	3×3	BB/BBL/TEM	2×7

The results of the experiments show that the best timing latency result is usually obtained with BB task mapping. This is because this mapping algorithm does not consider the latency constraints and traverses nearly all mapping possibilities to achieve the minimum energy consumption. For the same reason, its computation complexity is enormous and the implementation time of the algorithm is often far longer than with the other algorithms. Compared with BB, the TEM task mapping algorithm executes only one turn to get the mapping solution according to the bandwidth

and latency constraints. Its implementation time decreases dramatically but the timing latency of the corresponding task mapping scenarios will also be longer. The BBL task mapping algorithm is based on BB. Its implementation time is less than BB because it takes the latency constraints into account. The latency constraints provide more bounds to trim away the impossible patterns in advance. Compared with the BB and the TEM algorithms, the timing performance and the implementation time of BBL algorithm lie somewhere in between.

Like in the analysis of experiment #2, the 4×4 NoC offers more task mapping choices. The timing of the scenarios on the 4×4 NoC is usually better than on the other shapes. Its disadvantage is that it uses more FPGA resources. From the best solutions for multispectral imaging, the different results can be observed. The best solution with minimum dynamic energy consumption and the longest path latency is based on the 3×5 shape. The best solution for other requirements is based on the 2×7 shape. This result is mainly due to the task connection structure of this specific application.

As explained in the above discussion, the proposed task mapping exploration flow is vital for the designer, as it helps identify the appropriate task mapping technique on the suitable NoC shape for the specific application concerned. With the best task mapping strategy and the most suitable NoC shape, it is easier for the NoC designer to achieve good timing performance and low energy consumption.

For different real applications, especially image processing applications, an optimal mapping strategy can improve both timing performance and energy consumption. Likewise the appropriate NoC shape can reduce the resources used and further improve performance. Before the real implementation of a NoC design, selecting the appropriate task mapping algorithm and the NoC shape are vital for the design.

6. CONCLUSION AND PERSPECTIVES

This paper addresses an important issue in the NoC design: exploring the best possible task mapping and mesh topology. An exploration flow is proposed to help NoC designers to choose an appropriate task mapping technique on an appropriate NoC shape for a particular application. By combining the NoC configuration parameters, application communication parameters, and task mapping techniques, the flow generates different task mapping scenarios. By emulating these scenarios on the NoC emulation platform and analyzing the emulation results, the best solution can be found for a specific requirement. The complete exploration space is required for each algorithm. The experiments show that the exploration flow can help the designer to explore the best task mapping strategy to meet the design requirements of a particular application.

In this work, the latency constraints of the application task graph are used randomly. Our next task will be to explore the definition of the latency constraints of specific application. The adaptive definition of the latency constraints of specific application will help improve timing performance. It also will accelerate the task mapping execution time of BBL algorithm.

ACKNOWLEDGEMENTS

The research of this paper was supported by National High-tech Research and Development Projects (863) under Grant 2012AA012705, International Cooperation in Science and Technology Special Project of Ministry of Science and Technology of China under Grant 2012DFB10170, and China Scholarship Council under Grant 201306250116. And funding for this project was partly provided by a grant from la Région Rhône-Alpes as well as by the CNPq (process 245340/2012-2).

REFERENCES

[1] Ruxandra Pop and Shashi Kumar. A Survey of Techniques for Mapping and Scheduling Applications to Network on Chip Systems, Research Report, School of Engineering, Jönköping University, Sweden, 2004, pp. 1-9.

[2] Pradip Kumar Sahu and Santanu Chattopadhyay. A survey on application mapping strategies for Network-on-Chip design, Journal of System Achitecture 59(2013), 60-76.

[3] C.L. Chou, U.Y. Ogras and R. Marculescu. Energy- and performance-aware incremental mapping for NoCs with multiple voltage levels, IEEE Transactions on Computer-Aided design of Integrated Circuits and Systems 27 (10) (2008), 1866–1879.

[4] Carvalho, E. and Moraes, F.. Congestion aware Task Mapping in Heterogeneous MPSoCs, International Symposium on System-on-Chip, 2008. SOC, Tampere, Finland (2008), 1-4.

[5] T. Lei and S. Kumar. A two-step genetic algorithm for mapping task graphs to a network on chip architecture, Proceedings of the Euromicro Symposium on Digital System Design (DSD), Belek-Antalya, Turkey (2003), 180–187.

[6] J. Hu and R. Marculescu. Energy- and performance-aware mapping for regular NoC architectures, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24 (4) (2005), 551–562.

[7] X. Wang, M. Yang, Y. Jiang and P. Liu. Power-aware mapping for Network-on-Chip architectures under bandwidth and latency constraints, International Conference on Embedded and Multimedia Computing (EM-COM), Jeju, Korea (2009), 1–6.

[8] Carvalho, E., C. Marcon, N. Calazans and F. Moraes. Evaluation of Static and Dynamic Task Mapping Algorithms in NoC-Based MPSoCs, International Symposium on System-on-Chip, 2009. SOC, Tampere, Finland (2009), 87–90.

[9] Yangfan Liu, Peng Liu and Yingtao Jiang. Building a multi-FPGA-based emulation framework to support networks-on-chip design and verification, International Journal of Electronics - INT J ELECTRON 01/2010, 97(10) (2010), 1241-1262.

[10] J. Tan, V. Fresse and F. Rousseau. Generation of emulation platforms for NoC exploration on FPGA, 22nd IEEE International Symposium on Rapid System Prototyping (RSP), Karlsruhe (2011), 186-192.

[11] S. Lotlikar, V. Pai and P. Gratz. AcENoCs: a configurable HW/SW platform for FPGA accelerated NoC emulation. 24th Int. Conf. on VLSI Design, Chennai, India (2011), 147-152.

[12] D. Wang, N. E. Jerger and J. G. Steffan. DART: a programmable architecture for NoC simulation on FPGAs. Proc. of the Fifth ACM/IEEE Int. Symp. on NoC, NOCS ’11, New York, NY, USA (2011), 145–152.

[13] Y. Krasteva, F. Criado, E. de la Torre and T. Riesgo. A fast emulation based NoC prototyping framework. Int. Conf. on Reconfigurable Computing and FPGAs, Cancun, Mexico (2008), 211–216.

[14] J. Joven, O.F. Bach, D.C. Rufas, R. Martinez, L. Teres and J. Carrabina. xENoC – an experimental Network-on-Chip environment for parallel distributed computing on NoC-based MPSoC architecture, Euromicro Conference on Parallel, Distributed and Network-based Processing, Toulous, France (2008), 141–148.

[15] C. Ostler and K.S. Chatha. An ILP formulation for system-level application mapping on network processor architecture. Proceedings of Design, Automation and Test in Europe (DATE), ACM, Nice, France (2007), 1-6.

[16] M. Reshadi, A. Khademzadeh and A. Reza. Elixir: a new bandwidth-constrained mapping for networks-on-chip, IEICE Electronics Express 7 (2) (2010), 73–79.

[17] Krishnan Srinivasan and Karam S. Chatha. A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures, Proc. 2005 Int’l Symp. Low Power Electronics and Design (2005), 896-901.

[18] R. P. Dick, D. L. Rhodes and W. Wolf. TGFF: task graphs for free, Proc. Intl. Workshop on Hardware/Software Codesign (1998), 1-9.

[19] Carara, E. A.; Oliveira, R. P.; Calazans, N. L.V. and Moraes, F. G.. HeMPS - a framework

for NoC - based MPSoC generation, 2009 IEEE International Symposium on Circuit and Systems, ISCAS 2009, Taipei, Taiwan (2009), 1345- 1348.
[20] Otávio Alcântara de Lima Junior, Virginie Fresse and Frédéric Rousseau. Evaluation of SNMP-like protocol to manage a NoC emulation platform. International Conférence on Field Programmable Technology, 10-12 December 2014, Shanghai, China (2014).

AUTHORS

Ke Pang is a PhD student at Tianjin University, China. She obtained her B.Ss. and M.Sc. degrees at the School of Electronic Information and Engineering, Tianjin University. She studied at the Huber Curien Laboratory, Jean Monnet Univesity in Saint Etienne, France from 2013 to 2014. Her research interests include Network on Chip on FPGA, NoC design, embedded systems, image processing applications and the design of digital integrated circuits.

Dr. Virginie Fresse obtained her PhD in Electrical Engineering at
INSA Rennes, France, in 2001. Dr Virginie Fresse was in the Department Communication Division of the University of Strathclyde, Glasgow, from 2001 to 2003. She is currently associate professor at Jean Monnet University in Saint Etienne, France. Her research interests include Network OnChip on FPGA, emulation platforms, design space exploration and real time systems for image processing applications. She collaborates with French research labs (TIMA and GIPSA in Grenoble), and international labs (University of Tianjin, Pontifical Catholic University of Rio Grande do Sul (PUCRS) and Fortaleza and the University of Monastir).

Professor Suying YAO, professor and PhD. supervisor, is in charge of Tianjin University the key disciplines of Microelectronics and Solid State Electronics at Tianjin University, is director of Tianjin University ASIC Design Center, a committee member of Semiconductor and Integrated technology at the Chinese Institute of Electronics, and a member of IEEE.

Otavio Alcantara de Lima Jr. is assistant professor at the Federal Institute of Technology of Ceara, Brazil. He is also a PhD student in Micro-eletronics at Jean Monnet University, France. He received his MSc degree in Teleinformatics Engineering at the Federal University of Ceara, Brazil. His research interest includes embedded systems, MPSoCs and NoCs.