JAISCR, 2012, Vol. 2, No. 2, pp. 133 – 146

ADMINISTRATION-AND COMMUNICATION-AWARE IP CORE MAPPING IN SCALABLE MULTIPROCESSOR SYSTEM-ON-CHIPS VIA EVOLUTIONARY COMPUTING

Falko Guderian, Rainer Schaffer and Gerhard Fettweis
Vodafone Chair Mobile Communications Systems
Technische Universität Dresden, 01062 Dresden, Germany
e-mail:{falko.guderian, rainer.schaffer, fettweis}@ifn.et.tu-dresden.de

Abstract

In this paper, an efficient mapping of intellectual property (IP) cores onto a scalable multiprocessor system-on-chip with a k-ary 2-mesh network-on-chip is performed. The approach is to place more affine IP cores closer to each other reducing the number of traversed routers. Affinity describes the pairwise relationship between the IP cores quantified by an amount of exchanged communication or administration data. A genetic algorithm (GA) and a mixed-integer linear programming (MILP) solution use the affinity values in order to optimize the IP core mappings. The GA generates results faster and with a satisfactory quality relative to MILP. Realistic benchmark results demonstrate that a tradeoff between administration and communication affinity significantly improves application performance.

1 Introduction

Continuous technology scaling is driving an integration of many intellectual property (IP) blocks in a single chip allowing future generations of multiprocessor system-on-chip (MPSoC) to contain hundreds of heterogeneous processing elements (PEs) [1]. Network-on-chip (NoC) is a promising network design approach for scaling from MPSoC to Many-Core systems because the efficient communication infrastructure supports a large amount of PEs [2, 3]. As the two dimensional (2D) mesh scales poorly in NoC, a k-ary 2-mesh topology[1] with several IP cores at each router is proposed resulting in better performance while additionally improving area and energy efficiency [4].

The advantage of growing execution parallelism is faced with increasing dynamics and adaptability, e.g., from concurrency and dynamically changing application and user requirements. One way to cope with the challenge is to perform temporal and spacial resource allocation and task synchronization with a dedicated control processor (CP) dynamically at runtime. Resulting drawbacks from combining NoC and CP are potential limitations of data exchange. Assuming hundreds of cores, a large hop distance and contention will increase both administration and communication latency reducing application performance. To avoid such drawbacks, administration and communication should be considered together in system-level design.

Design space exploration (DSE) allows designers to explore and select designs at system-level. Given optimization goals and constraints, changing an application, architecture, and mapping characteristics result most likely in different design points. In our study, a tradeoff between administration and communication affinity aims at improved application performance through an adequate mapping of IP cores onto a NoC architecture. The design space in terms of possible placements is rapidly growing

[1] 2D mesh with k^2 routers located in k rows and k columns.

with NoC size. Moreover, several NoC design goals and constraints further increase design complexity. This motivates following heuristic solutions, allowing efficiency in finding adequate solutions. Genetic algorithms (GAs) have been proven to achieve this goal for over a decade. [5].

In this paper, we present an IP core mapping methodology and tool flow for MPSoCs which are scalable in terms of number of cores and the k-ary 2-mesh NoC topology. Communication and administration affinity are extracted from ideal application mappings of previously provisioned IP cores. Then, the approach considers a tradeoff between administration and communication affinity via linear weighting. A mixed integer linear programming (MILP) solution of the IP core mapping problem is provided as reference. A scalability analysis compares the performance of the proposed GA with the MILP. Realistic benchmark results demonstrate that the tradeoff is able to improve both administration and application performance.

In the remainder of the paper, Section 2 reviews related work. In Section 3, an overview of the IP core mapping approach is given. The GA will be introduced in Section 4. Then, Section 5 explains the MILP solution. We demonstrate experiments and results in Section 6. Finally, future work is discussed in Section 7. Section 8 concludes our work.

2 Related Work

In embedded system design, GAs have been applied to perform evolutionary computing for solving the IP core mapping problem. Existing work limits IP core mapping to k-ary 2-mesh NoC with one module per router so far. Ascia et al. [6] introduced a GA to perform multi-objective IP core mapping. Performance and power consumption have been balanced to obtain the Pareto mappings. The efficiency, accuracy, and scalability of the GA were shown for synthetic traffic and real applications.

Ozturk et al. [7] introduced heterogeneous NoC design based on a GA. Similar to us, an affinity matrix is used to describe communication between IP cores obtained through profiling. The authors present automatic selection of heterogeneous IP cores which are mapped afterwards, limiting NoC area. In addition, linear programming provides a reference for the GA.

Latif et al. [8] and Wang et al. [9] presented two GAs which significantly improve power consumption and system performance. In [8], IP cores with more communication requirements are given higher priority over less demanding IP cores. Afterwards, IP core mapping is performed according to a priority order. In our work, prioritization is also achieved by placing more affine cores closer to each other. In [9], tightly coupled application tasks will be located closer to each other compared to more distributed tasks. During the IP core mapping, power consumption is reduced, minimizing inter-core communications and considering both bandwidth and latency constraints. We also account for tightly / loosely coupled tasks because they imply more / less inter-core communication. We account for bandwidth and latency constraints during application mapping.

Our approach mainly differs from the related work because we separate communication and administration for solving the IP mapping problem. In addition, our study considers the k-ary 2-mesh NoC topology enabling efficient IP core mapping in future Many-Core systems.

3 IP Core Mapping Methodology

Figure 1 depicts an overview of our design methodology which maps IP cores to a k-ary 2-mesh NoC. Besides the optimization of an IP core mapping via a GA and MILP, the methodology requires further methods, such as simulation, estimation, (architecture) refinement, and validation. First, the IP cores are provisioned, such as via parallelism analysis, as introduced in [10]. Given the numbers and types of IP cores, the applications are dynamically mapped onto an architecture assuming an ideal point-to-point communication. The architecture further contains one or several memory and administration units. Due to the point-to-point communication between the IP cores, network delays and contention are avoided in the interconnect. From the application mapping results, an affinity value matrix A is extracted representing the pairwise relationship between the IP cores. Each element in A represents the communication and administration affinity between two IP cores. In the

next step, the optimization of the IP core mapping (placement) uses the affinity value matrix A. A GA-based solver, implemented in C++, and a MILP-based solver, implemented on a commercial tool, can be chosen for this task. Both optimization methods place highly affine cores at the same router and less affine cores will be located in a larger distance. Given the optimization results, the architecture is refined into a k-ary 2-mesh NoC. Then, the dynamic mapping of an application onto the NoC architecture is simulated. Afterwards, the mapping results are validated by a metric quantifying the system performance.

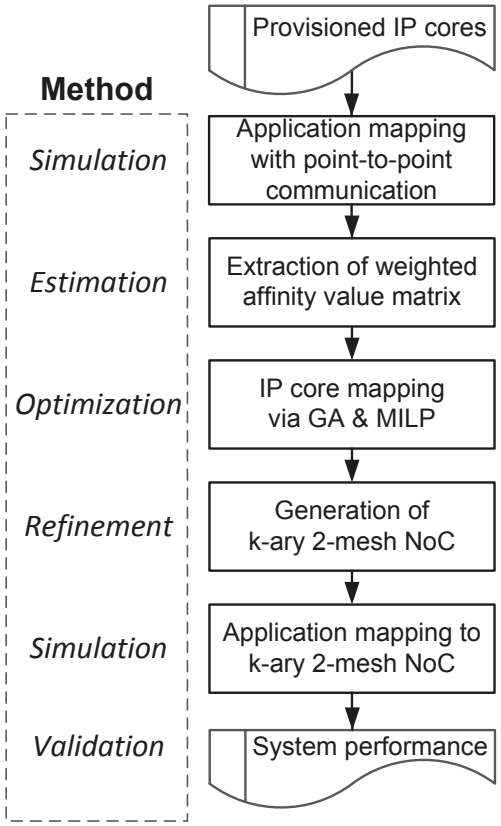

Figure 1. Overview of the IP core mapping methodology for a k-ary 2-mesh NoC.

3.1 Application and architecture model

In Figure 2, the architecture model describes a k-ary 2-mesh NoC with several IP cores per router resulting in better performance and power consumption compared to regular 2D mesh NoCs [4, 11, 12]. A k-ary 2-mesh NoC means that n routers are placed in a regular 2D mesh and each router connects several modules and IP cores, respectively. These include memory interfaces (Direct Memory Accesses, DMAs), CP interfaces (CP-IFs), and processing elements (PEs), such as general purpose processor (GPP), digital signal processor (DSP), application-specific integrated circuits (ASIC), etc. The set $\mathcal{C}$ comprises all IP cores. Moreover, $\mathcal{C}_{\mathcal{A}}$, $\mathcal{C}_{C}$, and $\mathcal{C}_{\mathcal{E}}$ define subsets of $\mathcal{C}$. $\mathcal{C}_{\mathcal{A}}$ contains the CP-IFs relating to administration and $\mathcal{C}_{C}$ includes the DMAs relating to communication. Moreover, $\mathcal{C}_{\mathcal{E}}$ consists of the PEs responsible for code execution. Concerning administration, a CP is responsible for dynamic task scheduling and resolving task dependencies at runtime. Several CP-IFs can be placed in a NoC to efficiently serve the PEs. The CP-IFs are connected to the CP through a dedicated network. The model also implies an application processor with dedicated memory access directly connected to the CP.

In the application model, threads represent applications which are independent on each other. Hence, both no data dependency and no control-flow dependency exist between threads. Each thread includes communication, administration, and computation tasks performed by the IP cores. Communication tasks are represented by data exchange between DMA and PE. Administration tasks, in our case initialization and release tasks, represent data transfer between CP-IF and PE. The initialization task models PE initialization before loading input data to the PE. After initialization and load task have finished, the PE executes the computation task. The release task models task synchronization after the computation result has been stored via DMA.

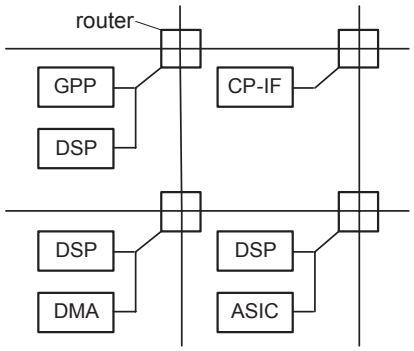

Figure 2. Architecture model with k-ary 2-mesh NoC and several IP cores per router.

3.2 Extracted affinity value matrix

IP core mapping considers affinity between cores in terms of exchanged communication and administration data. This requires application mappings created under ideal network conditions, as shown in Figure 1. The ideal application mappings consist of several task mappings. Given either communication or administration transfer between core i and core j, the task mappings allow to extract transfer latency $l_t(i,j)$, an amount of exchanged communication or administration data $d(i,j)$, and related execution latency $l_e(i)$ or $l_e(j)$. Transfer latency $l_t(i,j)$ is a time needed to transfer data $d(i,j)$ from core i to core j over a network. Execution latency $l_e(i)$ relates to code execution performed by PE i. The goal is to give faster cores higher priority over slower cores since they have more impact on the application performance. Therefore, $d(i,j)$ is weighted with the frequency of the data transfer, calculated via the ratio of $l_t(i,j)/l_e(i)$ and $l_t(i,j)/l_e(j)$, respectively. In case of a data exchange between two PEs, each PE contributes with half of its execution latency l_e. The weighted data values $d'(i,j)$ are derived for the cores i and j as follows:

$$d'(i,j) = \begin{cases} d(i,j) \cdot \frac{l_t(i,j)}{l_e(i)} & i \in \mathcal{C}_{\mathcal{E}}, j \notin \mathcal{C}_{\mathcal{E}} \\ d(i,j) \cdot \frac{l_t(i,j)}{l_e(j)} & j \in \mathcal{C}_{\mathcal{E}}, i \notin \mathcal{C}_{\mathcal{E}} \\ d(i,j) \cdot \frac{2\ l_t(i,j)}{l_e(i)+l_e(j)} & i \in \mathcal{C}_{\mathcal{E}}, j \in \mathcal{C}_{\mathcal{E}}, \\ & i \neq j \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In Equation 1, execution latency l_e relates to core i or core j which have to be an element of the set of computation cores $\mathcal{C}_{\mathcal{E}}$ and a PE type, respectively. Transfers between administration units (CP-IFs) and communication units (DMAs) are not considered. Due to several available task mappings between a core i and a core j, the weighted data values $d'(i,j)$ are accumulated over all tasks and included either in an affinity value matrix for communication A_C with cores $i \in \mathcal{C}_C \vee j \in \mathcal{C}_C$ or for administration A_A with cores $i \in \mathcal{C}_{\mathcal{A}} \vee j \in \mathcal{C}_{\mathcal{A}}$. Both matrices are normalized and weighted to be merged in an affinity value matrix A. Finally, the matrix is used as input for an IP core mapping, as illustrated in Figure 1.

In the following, a simple architecture containing three IP cores (DSP, CP-IF, DMA) is used to illustrate a tradeoff between communication and administration affinity. In the example, communication affinity is caused by memory data transferred between DSP and DMA and administration affinity relates to control data exchanged between DSP and CP-IF. Furthermore, DSP is responsible for code execution. The example below illustrates the two affinity value matrices A_C and A_A which have been extracted, normalized, and weighted to build an affinity value matrix $A = \omega \cdot A_C + (1-\omega) \cdot A_A$.

$$A = \underbrace{0.6}_{\omega} \cdot \underbrace{\begin{pmatrix} 0 & 0 & 0.6 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}}_{\text{communication}} + \underbrace{0.4}_{1-\omega} \cdot \underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 0.1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{administration}}$$

Accounting for execution latency, both matrices are either a representation of communication or administration data exchanged between the IP cores. In addition, weight ω is used to apply prioritization towards communication or administration affinity. Hence, the merged affinity value matrix A represents a tradeoff between communication and administration affinity. In the example, communication has been prioritized.

3.3 Focussed mapping objectives

In dynamically scheduled environments, it is necessary to initialize and synchronize task execution at run-time. As mentioned before, this is done by the CP-IF. Assuming an NoC, an increasing number of IP cores in future Many-Core systems will cause a larger average hop distance. Hence, administration latency between CP-IF and PE increases, reducing application performance. This makes it necessary to consider administration affinity in IP core mapping. We propose to weight administration and communication affinity in a linear fashion, see example before, to find a reasonable tradeoff aiming at improved application performance. Nevertheless, communication between PE and DMA represents a major network load which also needs to be considered for IP core mapping. In general, highly-affine cores are placed locally at the router and less-affine cores will be located in larger distance (#hops). We restrict our research to these design objectives and a single-objective function because we experienced that MILP solvers could only find an optimal solution for a reduced problem formulation and complexity, respectively.

As our work focuses on a tradeoff analysis, other optimization goals and constraints are left out for future work, see Section 7. Moreover, weighting has been modeled linearly both in the object function and the calculation of an affinity value matrix. The advantages include computational simplicity, an interpretable model form, and the ability to derive information of the quality of the fit. Another reason is that MILP does not efficiently support non-linear objective functions. Nevertheless, a non-linearly weighted affinity matrix is considered as future work.

3.4 Automated tool flow

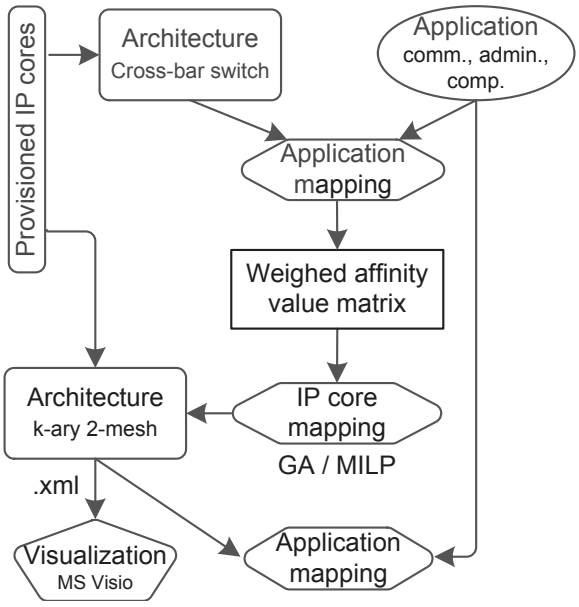

Figure 3. Automated tool flow for IP core mapping in k-ary 2-mesh NoC.

Figure 3 shows an automated tool flow for the IP core mapping. Given provisioned IP cores, a cross-bar switch NoC is generated representing ideal network conditions. Each application and thread, respectively, includes administration, communication, and computation tasks. The next step "Application Mapping" relates to scheduling and binding of tasks to cores performed by a CP dynamically at runtime. Application mappings for the ideal NoC and k-ary 2-mesh NoC will be derived from a DSE framework introduced in [13]. The framework has been extended to include NoC topology and protocols in the application, architecture, and mapping. Extraction of affinity value matrices was additionally realized. Earliest deadline first policy is used to schedule each thread and competing threads are prioritized via least laxity policy. Moreover, a thread will be canceled if a task deadline has been missed. Please note that extraction of affinity values and the IP core mapping are independent of the application mapping allowing also for different scheduling schemes. After application mapping to an ideal NoC, affinity value matrices for both communication and administration are extracted and merged to build a weighted affinity value matrix. Afterwards, the IP core mapping problem will be solved via GA or MILP. From the results, a suitable k-ary 2-mesh architecture is derived allowing to map applications onto the NoC. Furthermore, each architecture can be visualized using an XML representation and Microsoft Visio as graphical front end. Referring to Figure 3, the tool flow supports IP core mapping to be flexibly usable for an automated DSE approach. It allows to explore and compare several k-ary 2-mesh architectures using a weighted affinity matrix as starting point.

4 Evolutionary Framework

We developed a steady-state GA to evolve overlapping populations of individuals over a number of generations. The IP core mapping problem has been simply described by one-chromosome individuals. The chromosome is represented by a 3-dimensional array of genes. Two dimensions show the x-y position of a router. The remaining dimension maps IP cores to each router. Hence, each gene value represents an IP core. Following chromosome relates to the 2-ary 2-mesh NoC in Figure 2.

$$G = \begin{pmatrix} (\text{GPP DSP}) & (\text{CP-IF}) \\ (\text{DSP DMA}) & (\text{DSP ASIC}) \end{pmatrix}$$

Chromosome G includes four routers connecting one GPP, three DSPs, and one ASIC in addition to the DMA and CP-IF. Moreover, a fitness value represents the quality of each individual determined in the fitness rate (objective) function. Evolution starts with an initial population generated from pseudo-random choice. During each generation, individuals are selected and crossed according to the best fitness values. Then, new individuals are mutated by swapping genes according to a mutation rate. Variation through crossover and

mutation and subsequent selection allow to generate an improved offspring. One-point crossover can cause duplicated or missing IP cores and genes, respectively, resulting in illegal individuals. We introduce reparation after crossover by randomly replacing duplicated genes with missing genes. As shown in Section 6, this allows for a significantly improved fitness value compared to Monte Carlo simulation and a GA without reparation. The latter produces illegal individuals which have to be discarded during an optimization. From 100 IP cores, the reparation increases the simulation time only by around 1 % which is negligible.

In the GA, Algorithm 4 performs reparation of an illegal individual. The chromosome g contains genes g representing the IP cores placed in a NoC. Reparation is applied by randomly replacing duplicate IP cores (line 7) until all IP cores occur only once in the NoC (line 6: $\exists!$ g in g). Before each reparation, the duplicate genes are determined (line 3). In addition, the replacement of a duplicate IP core (line 7) is randomly initialized to improve the GA results. Lines 4 and 8 ensure that reparation proceeds until the individual gets legal.

Algorithm 1 Reparation of an illegal individual

```
1:  Select genome g
2:  while g is illegal do
3:      determineDublicateIPCores(g)
4:      g ← legal
5:      for each gene g in g do
6:          if not (∃! g in g) then
7:              randomlyReplaceIPCore(g)
8:              g ← illegal
9:              exit for
10:         end if
11:     end for
12: end while
```

Table 1 shows the constant terms used in our work for both the GA and MILP implementation. Assume that we are given N number of rows and M number of columns in a k-ary 2-mesh NoC, where $1 \leq x \leq N; 1 \leq y \leq M$. The NoC connects P IP cores, where $1 \leq i \leq P$.

The maximum amount of cores at router r is limited by $P_{\max}$ with $1 \leq r \leq R$. Affinity matrix values for core i and core j are expressed by $a_{ij} \in \mathbb{R}$ and they build an affinity value matrix $A \in \mathbb{R}^{P \times P}$. As mentioned in Section 3, weight ω with $0 \leq \omega \leq 1$ is used to linearly merge the affinity value matrices for communication A_C and administration A_A as follows:

$$A = \omega \cdot A_C + (1 - \omega) \cdot A_A. \tag{2}$$

Further weighting heuristics analysis are left for future research.

Table 1. The constant terms used in our GA/ MILP implementation. These are either architecture or application specific.

Constant	**Definition** [t]
N	number of rows in 2D mesh[t]
M	number of columns in 2D mesh [t]
P	total number of IP cores [t]
R	total number of routers [t]
$P_{\max}$	maximum IP cores connected to router [t]
a_{ij}	affinity weight between PE i and PE j (obtained as explained in Section 3) [t]
A	affinity matrix [t]
A_A	administration affinity matrix [t]
A_C	communication affinity matrix [t]
ω	linear weight for affinity matrices [t]
K	factor to prioritize local affinity [t]

The fitness value determines the best individuals from the population which are selected to generate a new population. The fitness rate (objective) function in Equation 3 maximizes the affinity values of closely placed IP cores.

$$\underbrace{\sum_{i=1}^{P}\sum_{j=1}^{P} b_{ij}a_{ij}}_{\text{global affinity}} + \underbrace{K \cdot \sum_{i=1}^{P}\sum_{j=1}^{P}\sum_{r=1}^{R} b_{ij}^{r}a_{ij}}_{\text{local affinity}} \rightarrow \max \tag{3}$$

The first term refers to global affinity which has been limited to neighbors connected to routers with one-hop distance. We use $b_{ij} = 1$ to indicate that core i and j are connected to routers with at least a distance of one hop. The second term describes local affinity of IP cores located at the same router and (x,y) position, respectively. Hence, $b_{ij}^{r} = 1$ denotes that both cores are located at the same router r. Weighting with factor $K > 1$ is used to prioritize local affinity higher than global affinity. Moreover, several distance are not considered keeping

the complexity of a MILP solution at a reasonable level.

GaLib [14] has been used for the GA implementation which overwrites the genetic operators and the fitness rate function.

So far, two different locations of the routers, local and global, are distinguished. Nevertheless, each router accounts for a latency and the effect on the transfer latency increases with the number of routers between two IP cores. This leads to an extension of the objective function by weighting the affinity value with a distance-dependent factor. The distance relates to the number of traversed routers. The new objective function is denoted as follows:

$$\sum_{i=1}^{P}\sum_{j=1}^{P}\frac{1}{\mathrm{dist}(i,j)}\cdot b_{ij}a_{ij} \rightarrow \max. \qquad (4)$$

The Equation 4 differs from Equation 3 in terms of the distance function $\mathrm{dist}(i,j)$ considering the local affinity and extending the global affinity. In case the cores i and j share the same router, the distance function returns $\mathrm{dist}(i,j)=1$. Each additional router increases $\mathrm{dist}(i,j)$ by an increment or factor.

4.1 Discussion

GAs are often criticized to rather converge to a local optimal than to the global optimal. Hence, it is important to maintain diversity in a population. In this paper, this is realized by following a random initalization and reparation. Another problem is a fitness rate function which is costly in terms of a large evaluation time. In that case, approximation methods help to reduce the evaluation complexity. For example, the pairwise affinity between the IP cores has been approximated with an affinity matrix. In this work, the solution quality of the developed GAs is sufficient. Nevertheless, another optimization algorithm may find a faster and/or better solution compared to a GA, such as simulated annealing, hill climbing, etc. This requires to obtain average performance metrics. However, it is out of the scope of this paper.

5 MILP Formulation

In linear programming (LP), a linear objective function is constrained via linear equalities and inequalities. Its feasible region is a convex polyhedron and an LP algorithm finds the smallest (or largest) value. In case all unknown variables are required to be integers, the problem is an integer LP problem. These problems are in many practical situations (those with bounded variables) non-polynomial hard. An mixed-integer LP (MILP) problem includes both unknown real and integers variables. The following MILP framework will be used as reference for the previously presented GA implementation. The problem is formulated in the LP format to be further solved by CPLEX [15] or other solvers. For the location of the P IP cores, we use the location matrix L with its elements l_{xy}. The location matrix L assigns each IP core a location in terms of an XY coordinate. The elements elements l_{xy} in L are calculated as follows:

$$\begin{aligned} l_{11} &= Q+1 \\ l_{x+1\ y} &= l_{xy}+1 \\ l_{x\ y+1} &= l_{xy}+Q \qquad Q>M. \end{aligned}$$

Q describes the step width in L that can be an arbitrary number with $Q>M,N$. Given $N<10$ rows and $M<10$ columns with $Q=10$, L is defined as follows:

$$L := \begin{pmatrix} 91 & \cdots & 99 \\ \vdots & \ddots & \vdots \\ 11 & \cdots & 19 \end{pmatrix} \quad N<10; M<10; Q=10.$$

Two location value l_{xy} are subtracted in order to determine either a northern, eastern, southern, and western neighborship. A set $\mathcal{M}$ includes the predefined distance metrics.

$$\mathcal{M} := \{\underbrace{+Q}_{\text{north}}, \underbrace{+1}_{\text{east}}, \underbrace{-Q}_{\text{south}}, \underbrace{-1}_{\text{west}}\}.$$

Given the IP core mappings, the neighborship of cores i and j is determined by comparing their distance in L with $\mathcal{M}$ as shown further below. The distance is taken from τ_{ij} representing the difference of the l_{xy} values according to the (x,y) position of cores i and j:

$$\tau_{ij} = \sum_{x=1}^{M}\sum_{y=1}^{N} l_{xy}b_{xy}^{j} - \sum_{x=1}^{M}\sum_{y=1}^{N} l_{xy}b_{xy}^{i} \quad \forall i,j \in P.$$

P defines the total number of IP cores. In the equation above, the location of core j at the (x,y) posi-

tion is given by the binary variable b_{xy}^{j}, more specifically:

$$b_{xy}^{j} := \begin{cases} 1 & \text{for the } j\text{th core placed at } (x,y) \text{ position} \\ 0 & \text{otherwise.} \end{cases}$$

An example illustrates the determination of a pairwise neighborship assuming the following locality matrix L with $Q = 10$:

$$L := \begin{pmatrix} \underline{21} & 22 \\ \mathbf{11} & 12 \end{pmatrix}.$$

It is given that a core (bold type number) is placed at a location corresponding to l_{11} and another core (underlined type number) corresponds to l_{21}. According to the set of distance metrics $\mathcal{M}$, $\tau_{12} = l_{21} - l_{11} = 10$ determines that the second core is the northern neighbor of the first core.

In addition, cores i and j can also share the same router r and (x,y) position, respectively, which is expressed by the binary variable b_{ij}^{r}:

$$b_{ij}^{r} := \begin{cases} 1 & \text{if cores } i \text{ and } j \text{ placed at same router } r \\ 0 & \text{otherwise.} \end{cases}$$

After describing all variables, we continue with the presentation of our constraints. The first constraint considers a unique placement of core j:

$$\sum_{x=1}^{M} \sum_{y=1}^{N} b_{xy}^{j} = 1 \quad \forall j \in P.$$

In the equation above, core j is limited to one (x,y) position. The k-ary 2-mesh topology allows for $P_{\max}$ IP cores at each router and (x,y) position, respectively. Hence, the following equation restricts the number of IP cores placed at each (x,y) position to $P_{\max}$:

$$\sum_{j=1}^{P} b_{xy}^{j} \leq P_{\max} \quad \forall x \in M; \forall y \in N.$$

Then, cores i and j are global neighbors if τ_{ij} is element of the distance metric set $\mathcal{M}$:

$$b_{ij} = \begin{cases} 1 & \text{if } \tau_{ij} \in \mathcal{M} \\ 0 & \text{otherwise.} \end{cases}$$

Assuming $P_{\max} > 1$, the following condition determines whether cores i and j share the same router r and (x,y) position, respectively:

$$b_{ij}^{r} = \begin{cases} 1 & \text{if } \tau_{ij} = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Given the necessary constraints in the MILP formulation, our objective function corresponds to the fitness rate function in Section 4 which maximizes the affinity values of closely placed IP cores.

6 Experimental Evaluation

First, the GA performance is evaluated using the MILP solution as reference. Processing times are normalized to a system with an AMD Opteron running at 2.2 GHz using one core. Second the reparation performance of the GA is compared with results from Monte Carlo simulations. Third, realistic benchmark results demonstrate the importance of determining ω to achieve a tradeoff between administration and communication affinity. GA parameters have been optimized based on the presented experimental setup. More specifically, 60 % crossover rate, 1 % mutation rate, and population size of 200 are choosen from these optimization results. Moreover, the two-distance fitness rate (objective) function, defined in Equation 3, is used.

6.1 Complexity scaling results for the GA and MILP implementation

We generated affinity value matrices A with pseudo-random values. The probability that two cores are affine in one direction has been set to 20 %. Problem size is scaled by increasing rows N, columns M, and maximum IP cores connected to routers $P_{\max}$ in the NoC. In each experiment, we run the GA for 10.000 generations and applied 100 reruns to account for the non-deterministic nature of the GA. Then, minimum, maximum, and average fitness values are extracted from the results. We set $K = 10$ to prioritize local affinity. In Figure 4, the performance behaviors of GA and MILP are depicted. For example, the MILP problem for the 2-ary 2-mesh with three IP cores per router includes 4121 linear constraints, 13 reals, 660 integers, and 2160 binaries. It was solved in around 2.14 hours. In contrast, GA executes each run with 10.000 generations in less than 1 second with an average deviation of 10 % related to MILP. Please note that using less generations implies faster optimization with

relatively low degradation of the deviation to MILP. In case of 48 IP cores, the GA solves the problem after 10.000 generations in around four seconds. Solution time for 1.000 generations is less than one second and the fitness value is 3.2 % smaller compared to the larger generation size. More figures are out of the scope of the paper.

Referring to Figure 4, a MILP solution was found for a maximum of 12 cores[2]. Instead, GA generates sufficiently high quality solutions in a fraction of time and the MILP solver can not even handle relatively small NoCs. The nature of a GA does not guarantee an optimal solution. For larger NoCs, the GA shows increasingly varying fitness values affecting the quality of solution.

In general, the solution time, when GA and MILP are converged, increases with the problem complexity. Whereas a larger sparsity of A decreases MILP problem complexity after variable optimization in the solver. For example with 12 and 75 cores, the GA converges after around 100 and 80.000 generations. Instead for 12 cores, the MILP problem already has thousands of variables and constraints, meaning a huge number of subproblems, solved by the branch & cut algorithm. Despite solution time also depends on the way how linear problems are described, it gives a reason why the MILP is not able to converge even for a smaller number of cores.

The GA promises to adequately solve larger problems than considered here. Regarding the quality of GA solutions, there is no further conclusion possible due to an absence of optimal MILP solutions for larger NoCs.

6.2 Comparison of GA w/ or w/o reparation and Monte Carlo simulation

In the experiments, the reparation performance of the GA is compared with Monte Carlo simulation by using the same amount of generated individuals. An absence of reparation has also been considered. For the GA, we assume the experimental setup of the previous section. The only difference is that the number of generations is set to 1000. The number of IP cores is increased from 10 to 100. Hence, the NoC topologies range from 3x3 to 5x5 with 3-4 modules per router ($P_{\max}$). The results are illustrated in Figure 5. Average (objective) fitness values have been generated by using 100 different seeds. In the figure, reparation improves the fitness value by around 40 % compared to the absence of reparation. Moreover, the reparation outperforms the Monte Carlo simulation by around 23 %. At 70 IP cores, a performance drop occurs for *Monte Carlo* and *No Repair GA*. The cause is that the NoC topology changed from 4x4 to 5x5 implying that 30 module interfaces are not used at the routers. Hence, both techniques can not efficiently solve the placement problem. Opposed to that, the impact on the performance of the reparation-based GA is much less. Moreover, reparation caused an average increase of 17 % in the solution time. The impact is decreasing with a larger number of IP cores and can be neglected for 100 IP cores (≈ 1 % increase).

6.3 Realistic benchmark results for a k-ary 2-mesh NoC topology

The multi-application scenario is derived from the E3S Benchmark Suite [16] which largely bases on data from the Embedded Microprocessor Benchmark Consortium [17]. It describes periodic task graphs and we use the auto-industry benchmark and the telecommunications benchmark due to their similar communication requirements. 95 concurrent threads are periodically executed and the number of threads is balanced amongst the two benchmarks. Considering diverse application starts t_s, t_s varies in the equally distributed time interval 0 cycles $\leq t_s \leq$ 10.000 cycles. The provisioned IP cores include 52 PEs from the PE types with the shortest task execution time. Together with six DMAs and six CP-IFs, the IP cores are mapped to a 5-ary 2-mesh NoC with maximum three IP cores per router. Unidirectional data exchange between cores occurs with ≈ 20 % probability. A_C and A_A have been increasingly weighted according to Equation 2 in the interval $\omega \in [0, 0.01, \ldots, 1]$. The non-deterministic nature of the GA requires averaging of results. Hence, 100 samples were recorded and averaged for each ω. A sample represents one GA run with 1.000 generations. Configuration and release tasks transfer 640 bit and 64 bit data. As mentioned in Section 3.4, application mappings are derived via simulation and NoC channel width is set to 64 bit. The k-ary 2-mesh NoC applies determin-

[2] otherwise aborts due to a 2 GB memory limitation

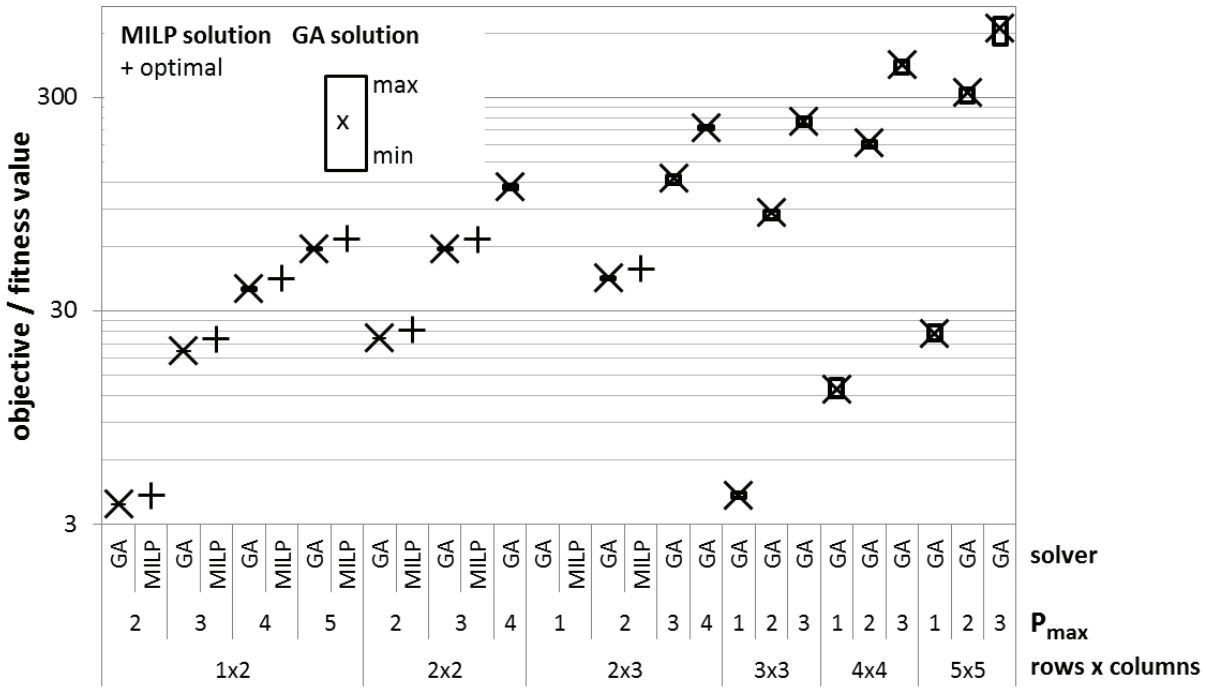

Figure 4. Complexity scaling results for MILP and GA.

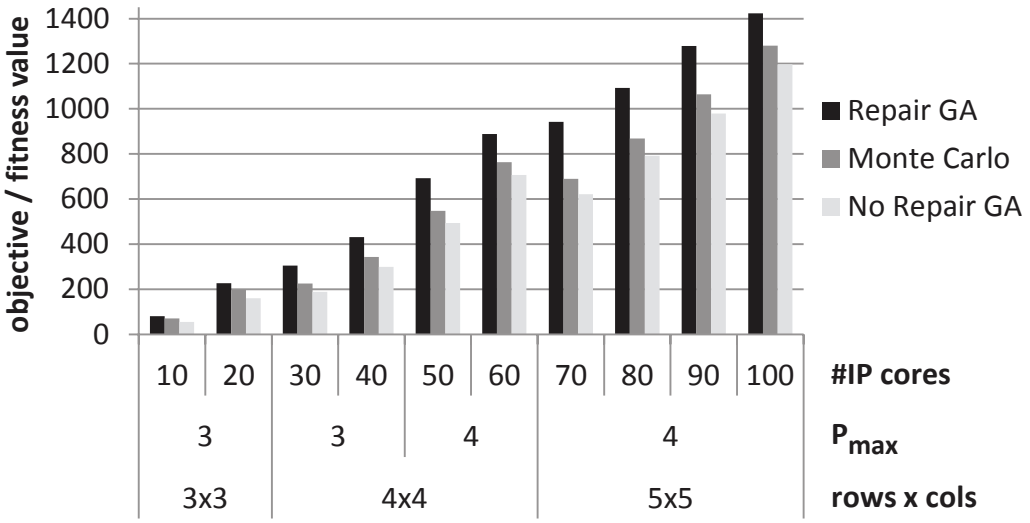

Figure 5. Comparison of GA w/ or w/o reparation and Monte Carlo simulation.

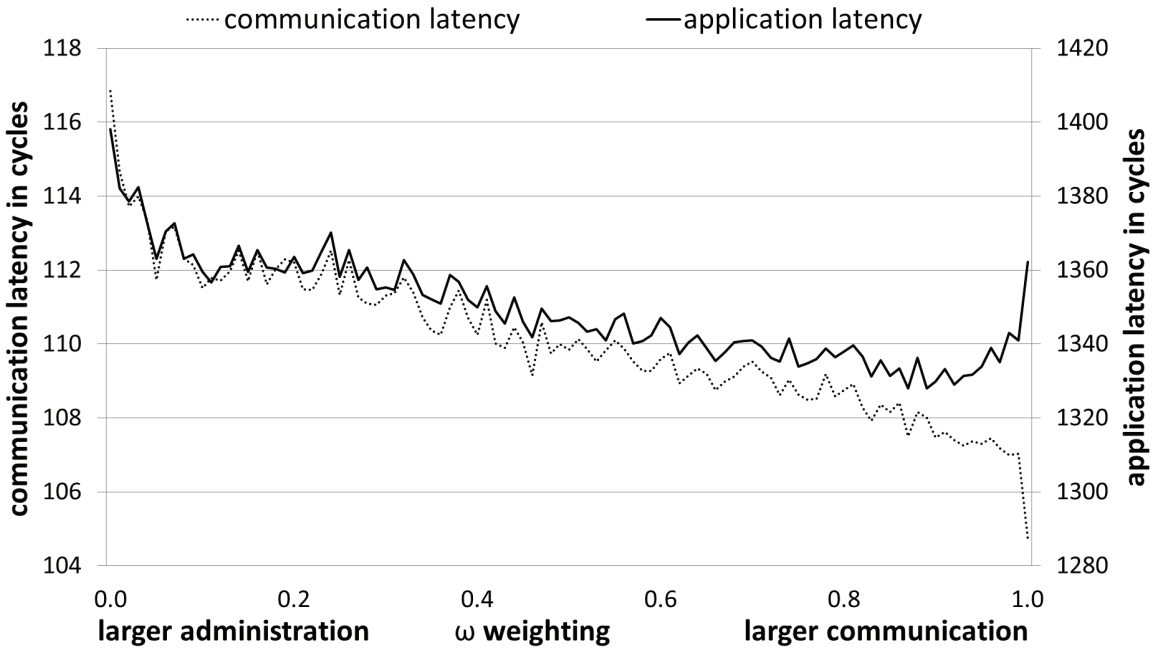

Figure 6. Benchmark results - Average application and communication latency for $\omega \in [0,1]$.

istic XY wormhole routing.

In Figure 6-7, the curves show average results of communication latency (l_C), administration latency (l_A), and application latency (l_{AP}) defined from a request time until an end of data transfer and execution, respectively. Referring to Figure 6, l_C decreases with increasing ω. This is because homogeneous placement of the DMAs is approached with larger ω. Despite homogeneous placement of the CP-IFs, l_A is not minimal for $\omega = 0$, as shown in Figure 7. This is because network load is dominated by communication. In the scenario, l_A finds its minimum for $\omega = 0.87$ with 17.5 % latency reduction compared to $\omega = 1$ where communication affinity is dominantly weighted. Hence, l_A is significantly reduced through the tradeoff between administration and communication affinity. Referring to Figure 6, l_{AP} has a minimum for $\omega = 0.87$ with 2.5 % improvement compared to $\omega = 1$. The slight l_{AP} reduction is due to the 3 % l_C increase for $\omega = 0.87$ compared to 17.5 % l_A reduction. Since l_A accounts only for around 20 % of the network load, l_{AP} improvement will be more significantly having more and larger administration data exchange. In addition, other scheduling schemes, e.g., through data locality-aware scheduling, will increase the impact of administration due to less communication data and tasks, respectively. From the results, we conclude that linear weighting allows for a tradeoff between administration and communication affinity improving application performance.

In the following, the results of the IP core mapping methodology are illustrated in terms of the location of the IP cores in the 5x5 NoC with maximum three IP cores at each router. The mapping corresponds to the best configuration using an affinity weight $\omega = 0.87$. This gives communication data a higher priority over the administration data. The result is shown in Figure 8. Therein, the communication and administration units are well distributed. A small distance from the computation units to the administration units (CP-IFs) and communication units (DMAs) significantly improves the system performance. Referring to Figure 8, the solution is not necessarily an optimal solution due to the heuristic nature of the GA. This configuration returns the lowest application latency (1295 cycles) of the 100 samples with $\omega = 0.87$.

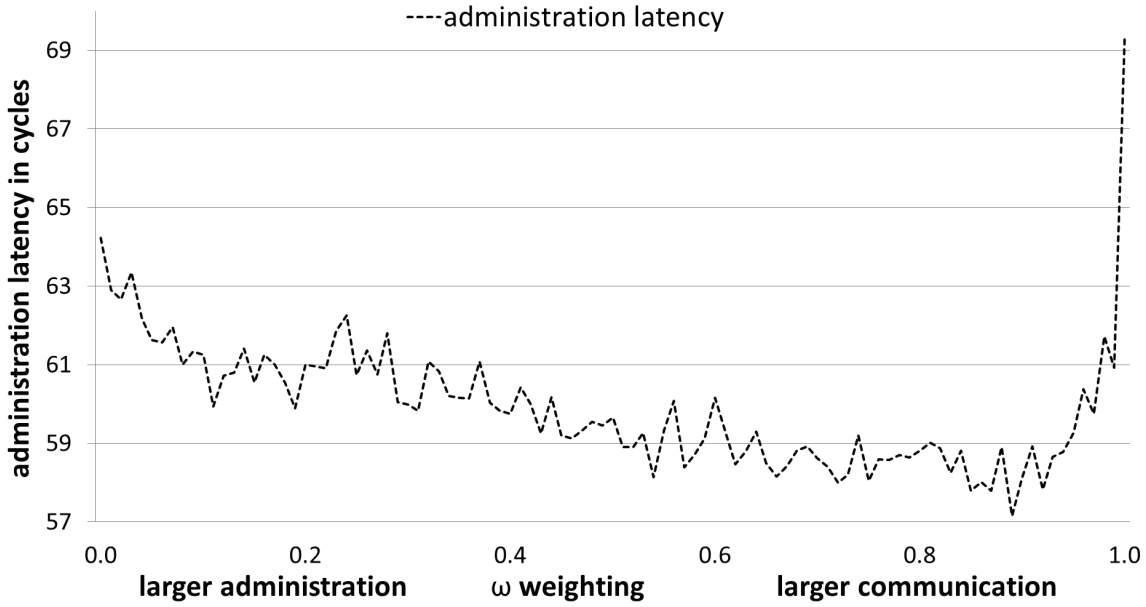

Figure 7. Benchmark results - Average administration latency for $\omega \in [0, 1]$.

7 Discussion and Future Work

In this paper, network load related to administration is limited to dynamic scheduling. Scenarios including more administration data exchange will further underline that administration affinity is becoming relevant to IP core mapping. Hence, future work should include additional network load, such as through monitoring of core temperature and NoC resources. In addition, management of voltage / frequency scaling increases data exchange related to administration. In these cases, a tradeoff between administration and communication affinity aims at a reduced administration latency to improve reliability, network performance, and power consumption. For example, energy-efficient systems and safety-critical applications require exchange of monitoring and control data over a network. Furthermore, exploitation of data locality relaxes network load related to communication. Concerning an optimization of IP core mapping, extensions should also address power, area and reliability goals, and constraints. Especially for the GA, it is fairly easy to integrate new design goals and constraints. In addition, it is necessary to extend the IP core mapping to irregular NoC topologies since a regular NoC used for heterogeneous cores would probably lead to inefficient area utilization. A possible solution requires to find a suitable geometric arrangement first and IP cores are assigned to routers afterwards. Hence, a new problem description of an IP core mapping has to be developed. Future work also includes an automatic DSE approach for a selection and mapping of IP cores similar to [7].

8 Conclusion

This paper presents IP core mapping for statically scalable MPSoCs in terms of number of IP cores and a k-ary 2-mesh NoC. First, affinity value matrices for communication and administration are extracted from ideal application mappings. Execution latency has been considered during extraction. Then, both matrices are linearly weighted to be merged into a single affinity value matrix. This enables a tradeoff between administration and communication affinity aiming at better system performance. The affinity value matrix is used as input for GA-based and MILP-based IP core mapping. A scalability analysis showed that the GA generates results faster and with a satisfactory quality relative

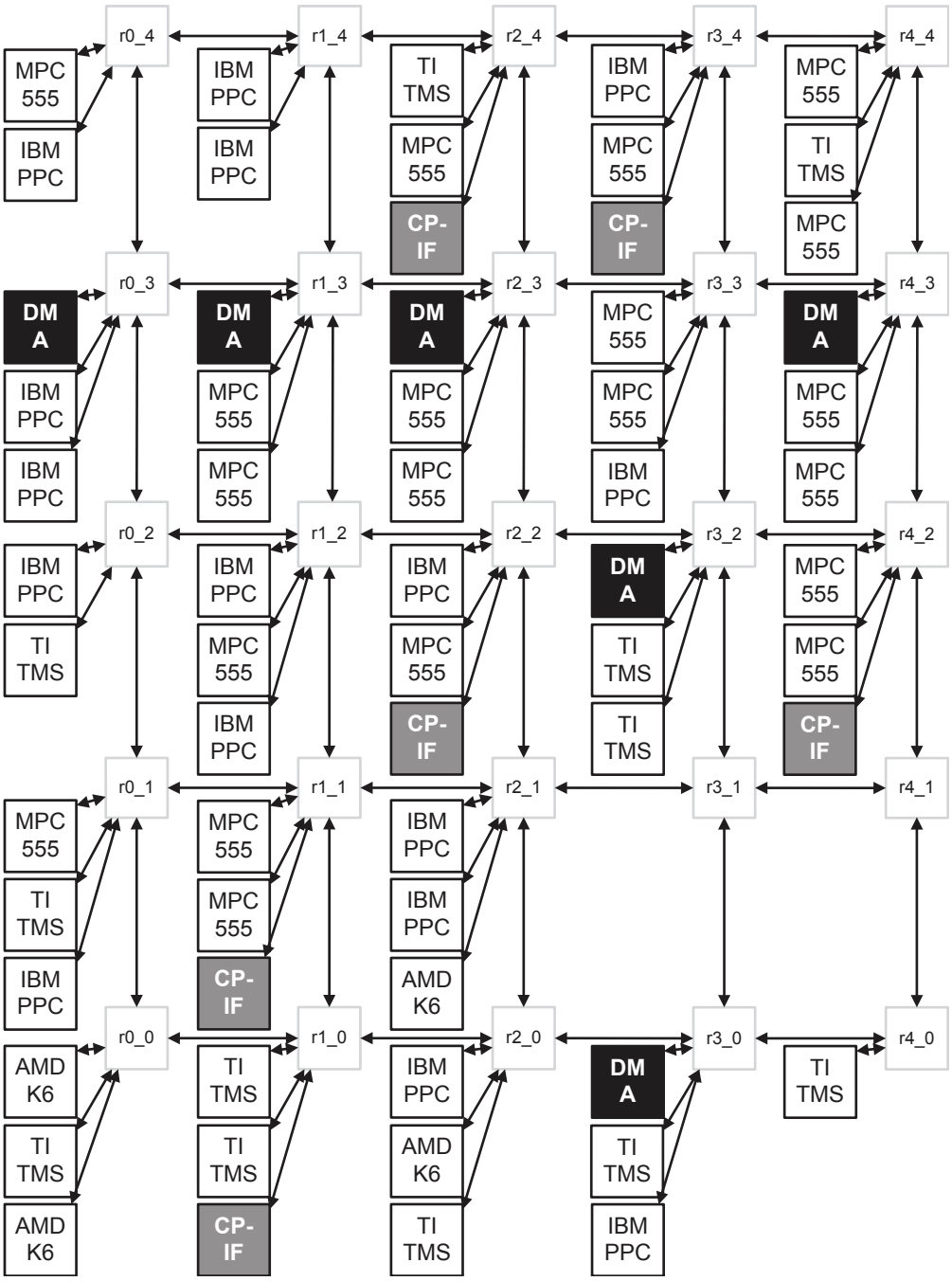

Figure 8. Best result configuration of the "IP core mapping" methodology

to the found MILP solutions. Moreover, the GA enables to find adequate solutions for more complex problems. Experiments with realistic benchmarks showed that a tradeoff between communication and administration affinity significantly reduces administration latency improving application performance. As network size and system adaptability increase, the growing influence of administration becomes more evident.

References

[1] S. Borkar, "Thousand core chips: a technology perspective," in *Proc. of DAC*, 2007, pp. 746–749.

[2] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proc. of DATE*, 2000, pp. 250–256.

[3] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Öberg, M. Millberg, and D. Lindquist, "Network on a chip: An architecture for billion transistor era," in *Proc. of NorChip*, 2000.

[4] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *Proc. of ICS*, 2006, pp. 187–198.

[5] H. Esbensen and E. Kuh, "Design space exploration using the genetic algorithm," in *Proc. of ISCAS*, 1996, pp. 500–503.

[6] G. Ascia, V. Catania, and M. Palesi, "Multi-objective mapping for mesh-based noc architectures," in *Proc. of CODES*, ser. CODES+ISSS, 2004, pp. 182–187.

[7] O. Ozturk and D. Demirbas, "Heterogeneous network-on-chip design through evolutionary computing," *International Journal of Electronics*, vol. 97, no. 10, pp. 1139–1161, 2010.

[8] K. Latif, A.-M. Rahmani, T. Seceleanu, and H. Tenhunen, "Power- and performance-aware ip mapping for noc-based mpsoc platforms," in *Proc. of ICECS*, dec. 2010, pp. 758–761.

[9] X. Wang, M. Yang, Y. Jiang, and P. Liu, "A power-aware mapping approach to map ip cores onto nocs under bandwidth and latency constraints," *ACM Trans. Archit. Code Optim.*, vol. 7, pp. 1:1–1:30, May 2010.

[10] B. Ristau, T. Limberg, O. Arnold, and G. Fettweis, "Dimensioning heterogeneous MPSoCs via parallelism analysis," in *Proc. of DATE*, 2009, pp. 554–557.

[11] C. Puttmann, J.-C. Niemann, M. Porrmann, and U. Ruckert, "Giganoc - a hierarchical network-on-chip for scalable chip-multiprocessors," in *Proc. of Euromicro*, 2007, pp. 495–502.

[12] F. Gilabert, S. Medardoni, D. Bertozzi, L. Benini, M. E. Gomez, P. Lopez, and J. Duato, "Exploring high-dimensional topologies for noc design through an integrated analysis and synthesis framework," in *Proc. of NOCS*, 2008, pp. 107–116.

[13] B. Ristau, T. Limberg, and G. Fettweis, "A mapping framework based on packing for design space exploration of heterogeneous mpsocs," *J. Signal Process. Syst.*, vol. 57, pp. 45–56, Oct 2009.

[14] M. Wall. (2011, Sep.) Galib: A c++ library of genetic algorithm components. [Online]. Available: http://lancet.mit.edu/ga/

[15] IBM. (2011, Sep.) Ibm ilog cplex optimizer. [Online]. Available: http://www.ibm.com/software/integration/optimization/cplex-optimizer/

[16] R. Dick. (2011, Sep.) Embedded system synthesis benchmarks suite. [Online]. Available: http://ziyang.eecs.umich.edu/∼dickrp/e3s/

[17] EEMBC. (2011, Sep.) The embedded microprocessor benchmark consortium. [Online]. Available: http://www.eembc.org/