Abstract

In this paper, we address the problem of an efficient mapping of intellectual property (IP) cores onto a multiprocessor system-on-chip (MPSoC). The MPSoC is statically scalable in terms of number of IP cores and an 1-ary n-mesh network-on-chip (NoC). The approach places more affine IP cores closer to each other and affinity is based on an amount of exchanged communication and administration data. Assuming ideal network conditions, accounting for execution latency, separate affinity value matrices for communication and administration are extracted from the application mappings. Aiming at better system performance, the goal is to find a reasonable tradeoff between communication and administration affinity. Hence, both matrices are merged into a single affinity value matrix based on linear weighting. A genetic algorithm (GA) and a mixed-integer linear programming (MILP) solution use the weighted affinity value matrix to efficiently map IP cores onto a NoC. A scalability analysis shows that the GA generates results faster and with a satisfactory quality relative to the found MILP solutions. Realistic benchmark results demonstrate that a tradeoff between administration and communication affinity significantly reduces administration latency improving application performance. As network size and system adaptability increase, the growing influence of administration becomes more evident.

Keywords IP core mapping · network-on-chip design · evolutionary computing · evolutionary design

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 n-mesh topology 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 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 1-ary n-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 1-ary n-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.

Öztürk 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 1-ary n-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 approach to map IP cores to an 1-ary n-mesh NoC. First, an ideal (zero delay) NoC is created from previously provisioned IP cores. The reason is to discard hop distance, router delay, protocol handling, and contention. From the results of application mappings onto an ideal NoC, an affinity matrix is extracted. Each affinity matrix value represents an amount of data transferred from one core to another and considers either administration or communication. The next step is IP core mapping using the affinity matrix. This is enabled through a GA-based solver implemented in C++ and a MILP-based solver implemented on a commercial tool. Both optimization methods place highly affine cores at the same router and less affine cores will be located in larger distance (#hops). Afterwards, an adequate 1-ary n-mesh NoC will be generated from the optimization result. The final architecture is used to map the applications onto the NoC.

3.1 Application and architecture model

In Figure 2, the architecture model describes an 1-ary n-mesh NoC with several IP cores per router resulting in better performance and power consumption compared to regular 2D mesh NoCs [4, 10, 11]. 1-ary n-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 \( C \) comprises all IP cores. Moreover, \( C_A \), \( C_C \), and \( C_E \) define subsets of \( C \). \( C_A \) contains the CP-IFs relating to administration and \( C_C \) includes the DMAs relating to communication. Moreover, \( C_E \) 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, repre-
sent 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.

### 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 \( l_{i,j} \), an amount of exchanged communication or administration data \( d_{i,j} \), and related execution latency \( e_{i,j} \) over a network. Execution latency \( e_{i,j} \) 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 a ratio of transfer latency \( l_{i,j} \) and execution latency \( e_{i,j} \) respectively. From the ideal task mappings, weighted communication and administration data values \( d'(i,j) \) are derived for cores \( i \) and \( j \) as follows:

\[
d'(i,j) = \begin{cases} 
  d(i,j) \cdot \frac{l_{i,j}}{e_{i,j}} & i \in \mathcal{C}_E, j \notin \mathcal{C}_E \\
  d(i,j) \cdot \frac{l_{i,j}}{e_{i,j}} & j \in \mathcal{C}_E, i \notin \mathcal{C}_E \\
  0 & \text{otherwise.}
\end{cases}
\]

In Equation 1, execution latency \( e_{i,j} \) relates either to core \( i \) or core \( j \) which have to be an element of \( \mathcal{C}_E \) and a PE, respectively. Hence, only transfers between PE and CP-IFs and between PE and DMAs are 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 \( i \in \mathcal{C}_C \cup j \in \mathcal{C}_C \) or for administration \( i \in \mathcal{C}_A \cup j \in \mathcal{C}_A \). 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 two affinity value matrices which have been extracted, normalized, and weighted to build an affinity value matrix \( A \).

\[
A = 0.6 \cdot \begin{pmatrix} 0 & 0 & 0.6 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 0.4 \cdot \begin{pmatrix} 0 & 1 & 0 \\ 0.1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}
\]

Accounting for execution latency, both matrices are either a representation of communication or administration data exchanged between the IP cores. In addition, weight \( \omega \) 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 Focused Mapping Objectives

In dynamically scheduled environments, it is necessary to initialize and synchronize task execution at runtime. As mentioned before, this is done by the CP. Assuming a 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 Section 4, 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 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 1-ary n-mesh NoC will be derived from a DSE framework introduced in [12]. 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 1-ary n-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 1-ary n-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 1-ary 4-mesh NoC in Figure 2.

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

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. This allows for significantly improved statistics of fitness values compared to no reparation is applied. Assuming a crossover rate of 60%, we use the realistic benchmark scenario in Section 6 as example. Variation of the fitness values improves by around 41% and the average fitness values increase by around 7%. In the example, reparation results in around 30% longer solution time. More results are left out for clarity reasons.

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 an 1-ary n-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 $P_{\text{max}}$ 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, $\omega \cdot 0 \leq \omega \leq 1$ is used to linearly merge the affinity value matrices for communication $A_C$ and administration $A_A$ as follows:

$$A = \omega \cdot A_C + (1 - \omega) \cdot A_A. \quad (2)$$

Further weighting heuristics analysis are left for future research. 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.

$$\sum_{j=1}^{P} \sum_{i=1}^{P} b_{ij}a_{ij} + K \cdot \sum_{i=1}^{P} \sum_{j=1}^{R} b_{ij}a_{ij} \rightarrow \max \quad (3)$$

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

<table>
<thead>
<tr>
<th>Constant</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>number of rows in 2D mesh</td>
</tr>
<tr>
<td>$M$</td>
<td>number of columns in 2D mesh</td>
</tr>
<tr>
<td>$P$</td>
<td>total number of IP cores</td>
</tr>
<tr>
<td>$R$</td>
<td>total number of routers</td>
</tr>
<tr>
<td>$P_{\text{max}}$</td>
<td>maximum IP cores connected to router</td>
</tr>
<tr>
<td>$\alpha_{ij}$</td>
<td>affinity weight between PE $i$ and PE $j$ (obtained as explained in Section 3)</td>
</tr>
<tr>
<td>$A$</td>
<td>affinity matrix</td>
</tr>
<tr>
<td>$A_A$</td>
<td>administration affinity matrix</td>
</tr>
<tr>
<td>$A_C$</td>
<td>communication affinity matrix</td>
</tr>
<tr>
<td>$\omega$</td>
<td>linear weight for affinity matrices</td>
</tr>
<tr>
<td>$K$</td>
<td>factor to prioritize local affinity</td>
</tr>
</tbody>
</table>

located at the same router and (x,y) position, respectively. Hence, $\beta_{ij}^{r} = 1$ 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. Including further distances would require an adequate weighting.

GaLib [13] has been used for the GA implementation which overwrites the genetic operators and the fitness rate function.

5 MILP Formulation

The MILP framework will be used as reference for the previously presented GA implementation. The problem is formulated in linear programming (LP) format to be further solved by CPLEX [14], a commercial tool. For the location of the $P$ IP cores, we use the location matrix $L$ with its elements $l_{xy}$. To determine the northern, eastern, southern, and western neighbors following rules for the values of element $l_{xy}$ are necessary:

$$l_{11} = Q + 1$$
$$l_{x+1,y} = l_{xy} + 1$$
$$l_{x,y+1} = l_{xy} + Q$$

$Q > M$.

Given $N < 10$ rows and $M < 10$ columns with $Q = 10$, $L$ is defined as follows:

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

Each entry in the set $M$ of distance metrics represents the neighborship of two cores either in north, east, south, or west direction. It is defined as follows:

$$M := \{+10, +1, -10, -1\} \text{.}$$

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

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

In the equation above, the location of core $j$ at the (x,y) position is given by the binary variable $\beta_{ij}^{xy}$, more specifically:

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

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 $\beta_{ij}^{r}$:

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

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

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

In the equation above, core $j$ is limited to one (x,y) position. The 1-ary n-mesh topology allows for $P_{\text{max}}$ 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 $P_{\text{max}}$:

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

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

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

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

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

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, realistic benchmark results demonstrate the importance of determining $\omega$ 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 have been chosen from these optimization results.

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 $P_{\text{max}}$ 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 1-ary 4-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 5, a MILP solution was found for a maximum of 12 cores (otherwise the solver aborts due to the 2 GB memory limitations). Instead, GA generates sufficiently high quality solutions in a fraction of time and the MILP solver can not even handle relatively small NoCs. For larger NoCs, the GA shows increasingly varying fitness values affecting the quality of solution. In general, the solution time, after GA and MILP converge, 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 smaller numbers of cores. Moreover, the nature of GAs does not guarantee an optimal solution. Nevertheless, 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 missing optimal MILP solutions for larger NoCs.

6.2 Realistic benchmark results for an 1-ary n-mesh NoC topology

The multi-application scenario is derived from the E3S Benchmark Suite [15] which largely bases on data from the Embedded Microprocessor Benchmark Consortium [16]. 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 $t_\omega$, $t_\lambda$ varies in the equally distributed time interval 0 cycles $\leq t_\omega \leq 10,000$ 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 1-ary 25-mesh NoC with maximum three IP cores per router. Unidirectional data exchange between cores occurs with $\omega = 20$ % probability. $A_\omega$ and $A_\lambda$ 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 $\omega$. 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 1-ary n-mesh NoC applies deterministic XY wormhole routing.

In Figure 5-6, the curves show average results of communication latency (CL), administration latency (AL), and application latency (APL) defined from a request time until an end of data transfer and execution, respectively. Referring to Figure 5, CL decreases with increasing $\omega$. This is because homogeneous placement of the DMAs is approached with larger $\omega$. Despite homogeneous placement of the CP-IFs, AL is not minimal for $\omega = 0$, as shown in Figure 6. This is because network load is dominated by communication. In the scenario, AL finds its minimum for $\omega = 0.87$ with 17.5 % latency reduction compared to $\omega = 1$ where communication affinity is dominantly weighted. Hence, AL is significantly reduced through the tradeoff between administration and communication affinity. Referring to Figure 5, APL has a minimum for $\omega = 0.87$ with 2.5 % improvement compared to $\omega = 1$. The slight APL reduction is due to the 3 % CL increase for $\omega = 0.87$ compared to 17.5 % AL reduction. Since AL accounts only for around 20 % of the network load, APL 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...
Figure 4: Complexity scaling results for MILP and GA.

Figure 5: Benchmark results - Average application and communication latency for $\omega \in [0, 1]$.

Figure 6: 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 trade-off 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 an l-ary n-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 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