



Article

Application-Specific SoC Design Using Core Mapping to 3D Mesh NoCs with Nonlinear Area Optimization and Simulated Annealing

Jan Moritz Joseph ^{1,*} , Dominik Ermel ¹, Lennart Bamberg ², Alberto García-Ortiz ² and Thilo Pionteck ¹ 

¹ Institut für Informations- und Kommunikationstechnik, Otto-von-Guericke-Universität Magdeburg, 39106 Magdeburg, Germany; dominik.ermel@ovgu.de (D.E.); thilo.pionteck@ovgu.de (T.P.)

² Institute of Electrodynamics and Microelectronics, University of Bremen, 28359 Bremen, Germany; bamberg@item.uni-bremen.de (L.B.); agarcia@item.uni-bremen.de (A.G.-O.)

* Correspondence: jan.joseph@ovgu.de

Received: 26 November 2019; Accepted: 21 January 2020; Published: 23 January 2020



Abstract: Core mapping, in which a core graph is mapped to a network graph to minimize communication, is a common design problem for Systems-on-Chip interconnected by a Network-on-Chip. In conventional multiprocessors, this mapping is area-agnostic as the cores in the core graph are uniform and therefore iso-area. This changes for Systems-on-Chip because tasks are mapped to specific blocks and not general-purpose cores. Thus, the area of these specific cores is varying. This requires novel mapping methods. In this paper, we propose an area-aware cost function for simulated annealing; Furthermore, we advocate the use of nonlinear models as the area is nonlinear: A semi-definite program (SDP) can be used as it is sufficiently fast and shows 20% better area than conventional linear models. Our cost function allows for up to 16.4% better area, 2% better communication (bandwidth times hop distance) and 13.8% better total bandwidth in the network in comparison to the standard approach that accounts for both the network communication and uses cores with varying areas as well.

Keywords: Network-on-Chip; core mapping

1. Introduction

Core mapping is one important design-time optimization problem for chips interconnected by Network-on-Chips (NoCs). The target of this mapping problem is a better distribution of work among the cores to improve data movement between them. Different objectives can be found in the literature such as reducing power [1] or avoiding bandwidth limitations [2]. In this paper, the objective optimizes the area of the chip, which is not commonly found in the literature so far because of the tacit assumption of iso-area cores. However, this is not valid for all systems, as specific blocks will have different areas in contrast to general-purpose cores. We extend our work from [3], in which we proposed a nonlinear model to improve area, by incorporating this model in a simulated annealing cost function to solve area-aware core mapping.

The problem of core mapping is defined as follows: The application's data streams are modeled using a core graph, in which nodes represent cores and edges with their edge weight model the bandwidth of data stream between the cores. This core graph is mapped to a chip, typically a multiprocessor interconnected by an NoC. The chip is represented by a network graph, in which nodes model tiles that reserve space for an NoC router and a core, and edges model links between tiles. The objective of this optimization is minimization of network latency, typically measured

in the cumulative hop distance \times bandwidth (e.g., [2]), maximization of the throughput, typically measured by the maximum bandwidth transmitted through single links (e.g., [4]), minimization of energy consumption, measured in dynamic router activity (e.g., [1]) or minimization of execution time, measured by the hop distance along the critical path (e.g., [5]). Core mapping is a very common electronic design automation (EDA) task in NoC design and many approaches from exact analytical solutions, e.g., [5], to heuristics such as simulated annealing (SA), e.g., [2], have been proposed.

Recently, Systems-on-Chip (SoCs) gained attention. There are two important differences to multicore processors: First, the function and thus the area of cores varies in SoCs. Thus, the underlying assumption of iso-area cores, which lead to disregarding the core area during core mapping, is not valid anymore. Second, many SoCs such as vision chips [6] are application specific, while multiprocessors are not. Thus, the application properties must be accounted for already during core placement to exploit additional optimization potential. Using conventional approaches, each tile would have to reserve area for the largest core, which is inefficient, naturally. An example is depicted in Figure 1, in which cores of different sizes (orange) allocate less area than reserved (light gray). Therefore, novel approaches are required.

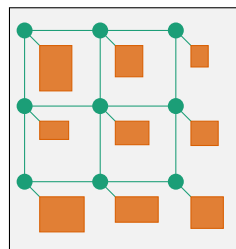


Figure 1. Cores (orange) of different areas on an SoC, attached to a 2D-mesh NoC [3].

Ref. [3] introduced nonlinear models and compared them against linear models to optimize area during core mapping. Here, we extend this work by proposing a cost function for simulated annealing to optimize the core mapping and the chip area; Specifically, the nonlinear models are used to optimize area within the simulated annealing. Since these nonlinear models are exact for area, we propose a mixed-exact-approximate method to minimize communication and area in SoCs during design-time core mapping.

The remainder of this work is structured as follows: A review of the state-of-the-art is given in Section 2. Next, the area-aware simulated annealing is introduced in Section 3 that uses linear or nonlinear models to optimize area. Both of these models are introduced in Section 4, which are based on our preliminary paper [3] that this work extends. The results obtained with the simulated annealing are reported in Section 5. The work is concluded in Section 6.

2. Related Work

As already explained, many works on core mapping exist. In general, there are two classes of mapping methods: *Exact approaches* using an analytical model such as a mixed-integer linear program (MILP), an extensive search of the solution space or *heuristic approaches* that approximate the solution at better runtime, e.g., simulated annealing or particle swarm optimization. Within each class, the approaches can be further classified by their objective functions that e.g., minimize power or maximize performance.

In the first class of *exact approaches*, the vast majority of works use mixed integer linear programming to solve core mapping: Ref. [7] allows for connecting multiple cores to a single router. By that, energy consumption is minimized by up to 81.2% compared to one-to-one connections. Ref. [8] optimizes mapping and topology selection to minimize bandwidth, area and network component savings by a minimum of 50% each in comparison to traditional design approaches. Ref. [9] maximizes the worst case throughput and also accounts for multi-threaded processes, i.e., mapping of multiple cores to a single tile in the network graph. Ref. [10] minimizes communication energy using mapping;

it targets integration into frameworks that find optimal network voltage and frequency. In particular, Ref. [2] is worth mentioning as it is the standard work core mapping in that cores have different areas. The work uses MILP to synthesize an NoC topology from a core graph with area and power annotations. In contrast to this paper, area reduction is not objective. Rather, Ref. [2] reduces power. Different multimedia benchmarks [11] are used for evaluation. This paper is closely related to [2] because of the consideration of area. Thus, we compare against the same benchmarks for a fair comparison. As our objective function is different, we are able to achieve better area figures.

In the second class of *heuristic approaches*, many different algorithms have been explored, e.g., genetic algorithms (GA), in which solutions evolve, particle swarm optimization (PSO), in which agents collaboratively find a good solution, or simulated annealing (SA), in which cooling processes are used as inspiration to find optimal configurations. Ref. [12] uses GA to minimize the overall execution time of the application. More recently, GA is rarely used due to lower runtime than PSO and SA: Ref. [13] optimizes mapping for partially vertically-connected 3D NoCs to make best use of through-silicon via (TSVs). The authors propose both and MILP and a PSO to improve network congestion, but the MILP has too long of a runtime for realistic use cases. SA is one of the most-used EDA methods. It can be used at many abstraction levels, from gate-level [14] to system-level optimization [15]. A reason therefore lies in SA's compelling performance, as we will show with a comparison against PSO in this work. The performance for SA can be further by combination with different techniques: For example, Ref. [16] shows that SA core mapping combined with cluster analysis allows for up to 30% better runtime at the same quality of results in comparison to off-the-shelf SA. Ref. [17] shows that application of further knowledge about the structure of the objective function allows for up to 66% better average energy consumption in comparison to a blind search. Ref. [18] also focuses on reduction of energy consumption. In this approach, the router allocation is done prior to voltage islanding, thus saving up to 63% power and delay over Sunfloor [1]. Ref. [19] focuses on runtime reductions under thermal constraints, which are specifically challenging in 3D NoCs. In their work, the authors formulate a communication and thermal aware mapping problem and solve it using custom heuristics. They achieve up to 43% better runtime than related works.

To summarize, there are many approaches in the literature on core mapping in NoC-based multiprocessors. Only a small subset accounts for area because most of the works assume homogeneous cores, which is not always valid for heterogeneous scenarios. As area is nonlinear, intrinsically, it is not possible to model it exact through means of linear models. Thus, a novel approach is required as proposed in this work.

3. Area-Aware Core Mapping with Simulated Annealing

3.1. Problem Definition

The problem of core mapping has been defined multiple times, e.g., [16,20]. The difference to our definition lies in the annotation of the core graph with area. Even more, this area annotation enables the definition of a new objective function including area. The problem of core mapping takes a core graph and a network as input. These are defined as follows:

Definition 1 (Core Graph). *The core graph models the area of cores as well as the bandwidth requirements for communication between cores. It is digraph $CG = (C, E_C)$, in which the set C of vertices consists of all cores c_i , with $i \in \{1, \dots, |C|\}$ as the set of core indexes. The set of directed edges $e_{i,j} \in E_C$ models the communication between cores c_i and $c_j \in C$. Cores are area-annotated by the function $area : C \rightarrow \mathbb{R}^+$. The bandwidth between nodes is given by the capacity function $bandwidth : E_C \rightarrow \mathbb{R}^+$.*

Definition 2 (Network Graph). *The network graph models the interconnection topology of the set of target SoC architectures. The network graph is a undirected graph $NG = (T, E_T)$, in which the set T of vertices consists of tiles t_i , with $i \in \{1, \dots, |T|\}$ the set of tile indexes, which implement one NoC router each and reserve*

space for the area of a mapped core. The set of edges $e_{i,j} = e_{j,i} \in E_T$ models the connections between routers in tiles t_i and $t_j \in T$.

The aim of the core mapping is to find a mapping that minimizes an objective function.

Definition 3 (Mapping Function). *The mapping function assigns a core $c \in C$ to a tile $t \in T$. It is defined as*

$$\text{map} : C \rightarrow T.$$

The mapping function is injective because each tile can only host one core.

We also define two auxiliary functions. First, the mapping of cores to tiles results in an area requirement for each tile:

Definition 4 (Network Area Function). *The area requirement of each tile is given by the function*

$$F : T \rightarrow \mathbb{R}^+$$

that is defined as $F(t_j) = \text{area}(\text{map}^{-1}(t_j))$. Since the mapping function is injective, map^{-1} is well-defined on the image set of map . Where $\text{map}^{-1}(t_j)$ is not defined, we define $F(t_j) = 0$ instead.

Second, we model the *flow* of packets in the network graph, i.e., the paths of packets based on the routing algorithm, as a source-sink-flow in the network digraph.

Definition 5. *We define the function f that gives the network flow for each pair of components:*

$$f : E_T \rightarrow \bigcup_{(c_i, c_j) \in E_C} \{h \mid h \text{ is a } c_i\text{-}c_j\text{-flow in } NG, \text{value}(h) = 1\}. \quad (1)$$

The value of the $c_i\text{-}c_j\text{-flow}$ is denoted by $\text{value}(f)$, following the convention used in [21]. Flows are a very powerful concept, as they give a natural approach to the conversion of core flows to network flows. The function f assigns a flow value to every edge in the network graph E_T by considering the flow induced by all edges in the core graph E_C . Hence, a flow for a specific pair of cores is assigned to all links in the network, which will be passed by its packets. Consequently, both deterministic and adaptive routing algorithms can be modeled. As packets following deterministic routing algorithms only have one path through the network, the values of the flows will be binary, i.e., the set of links passed by packets has a flow value of 1, while all other links have a flow value of 0. In case of adaptive routing algorithms, the values of the flow along each link will be in the interval of $[0,1]$. This flow value represents the probability that a packet from the pair of cores will pass this very link when routed.

By that, the objective function of the area-aware core mapping can be defined:

Definition 6 (Objective Function).

$$O = w_1 O_{\text{area}} + w_2 O_{\text{latency}} + w_3 O_{\text{bandwidth}}. \quad (2)$$

The heightened addends are defined as follows: The costs for area O_{area} are the total area of the chip including whitespace based in F for a given mapping. The calculation of area is strictly dependent on the network topology. Here, we use a 2D mesh, as shown in Figure 1. The mesh reduces the spacial freedom and requires that cores are located in a grid. Thus, the area of the chip is given by the width W and the height H of the floorplan for this mapping. As the area WH is a product and difficult to calculate in linear models, we use the easy-to-linearize

maximum function to approximate area. This results in a model favouring squared chips. The objective for the chip area thus is:

$$O_{area} = \max(W, H).$$

The costs for latency are measured by the hop distance \times bandwidth for all data streams in the core graph:

$$C_{latency} = \sum_{c \in E_C} \text{bandwidth}(c) |f(c)|.$$

The costs for bandwidth are the maximum bandwidth of any link in the network graph for a given mapping:

$$O_{bandwidth} = \max_{v \in E_T} \sum_{c \in E_C} \text{bandwidth}(c) (f(c)(v)).$$

This objective function defines the novel problem of area-aware core mapping.

3.2. Simulated Annealing

We solve the area-aware core mapping using simulated annealing. We also implemented an exact solution using an MILP. Since the runtime of this MILP is very poor and only allows for solving input sets with up to seven components in reasonable time, we do not give a detailed definition. Therefore, a heuristic such as a simulated annealing is required. The steps of the simulated annealing are shown in Figure 2. An initial mapping is calculated from a core graph CG and a network graph NG . As depicted, this mapping does not optimize area and therefore includes whitespace (here shown in gray), i.e., unused die area. The initial mapping can either be random or area-efficient, depending on the goals of the optimization. Next, the simulated annealing is executed. The algorithm is initialized with this given valid, but possibly inefficient solution. The solution candidate is modified iteratively by executing the neighbor function that slightly changes the mapping map . As a novel feature, we optimize the area analytically within the simulated annealing before calculating the objective function (“Minimize Area” in Figure 2). This allows for a precise area optimization beyond the limitations of heuristics approaches possible by the simulated annealing. We will explain the analytical optimization of area in a separate Section 4. After this analytical step, the complete objective is calculated in that step of the simulated annealing. The algorithm might accept the novel solution based on the value of the objective function. Naturally, the simulated annealing is iterated and stopped when the terminating conditions are met. The final solution returns a mapping function map that minimizes area and communication, i.e., includes a floorplan for the given mapping. The initial solution and the neighbor function of the simulated annealing are defined as follows:

Definition 7 (Initial Solution). *There are two ways to generate an initial solution:*

1. **Randomly generated:** *The function map is generated such that each core c_i is assigned to one random tile t_j .*
2. **Area-efficient:** *The floorplan will be packed area-efficiently, i.e., with minimal whitespace, if all tiles within a row and a column have a similar area. Such a good candidate can be found using a greedy strategy: The cores are sorted descending by area. The tiles are filled from the upper left corner. The cores are assigned to the next free tile in the current row or column, while row and column assignment are alternating. If a row/column is full, tiles will be assigned to the adjacent one. Figuratively speaking, the tiles are filled from the upper left corner to the bottom right corner.*

Definition 8 (Neighbor Function). *The neighbor function (or move function) modifies a given mapping function map such that it takes the function map as input and returns a modified function map' . Specifically, it is modified by selection a core c_i and a tile $t_j \neq map(c_i)$ uniform randomly. The selected core is mapped to the selected tile, i.e., position. If a core is already present there, the two mapped cores are swapped. Thus, the modified mapping function map' is defined as follows:*

$$\text{map}'(c) = \begin{cases} t_j & \text{for } c = c_i, \\ \text{map}(c_i) & \text{for } \text{map}^{-1}(t_j) \text{ is defined and } c = \text{map}^{-1}(t_j), \\ \text{map}(c) & \text{else.} \end{cases} \quad (3)$$

This concludes the definition of the simulated annealing. It remains to optimize the area for a given mapping map , as explained in the next section.

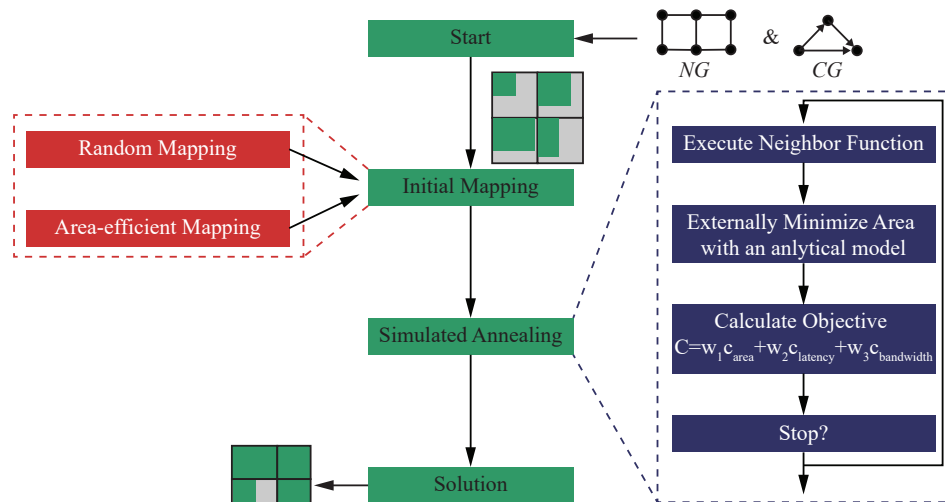


Figure 2. Simulated annealing algorithm.

4. Area Optimization for a Given Mapping Using Linear and Nonlinear Models

Conventionally, a mapping of tasks to (multiprocessor) cores would not require an optimization of area since cores are assumed to be identical and thus equal in area. However, this assumption will not be valid in SoCs since each task implements a different IP with varying size. Hence, tasks must also reserve adequate area for their implementing IP. The optimization problem's constraints and variables are shown in Figure 3: Cores are mapped to a mesh of tiles. Each core has an area value, denoted by $F_{i,j}$ for a core in a given row i and column j . The height of all rows and width of all columns, denoted by c_i and r_j respectively must be minimized. The multiplication of width and height is constrained by the size of the mapped cores. By that, the white space (gray in Figure 3) is reduced.

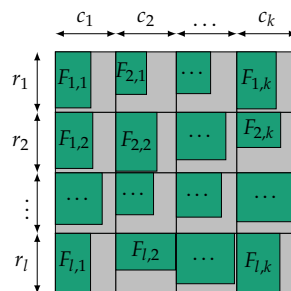


Figure 3. Variables and constraints of area optimization.

More specifically, the area optimization problem during core mapping is formulated as follows [3]: Assume a given mapping of less than or equal of lk cores to tiles in a mesh of l rows and k columns. Each core has the area $F_{i,j}$, for mapping to row $i \in [l] := \{1, \dots, l\}$ and column $j \in [k] := \{1, \dots, k\}$. For all empty tiles without a mapped core, $F_{i,j} = 0$ will be zero. The height of rows

is denoted by $r_i \in \mathbb{R}$ for all $i \in [l]$. The width of columns is denoted by $c_j \in \mathbb{R}$ for all $j \in [k]$. The area of each tile is constrained by its mapped core:

$$r_i c_j \geq F_{i,j} \quad \text{for all } i \in [l], j \in [k]. \tag{4}$$

The objective function O_2 minimizes the side length of a square that encloses all tiles (i.e., $W = \sum_{j \in [k]} c_j, H = \sum_{i \in [l]} r_i$):

$$O_2 = \max \left(\sum_{i \in [l]} r_i, \sum_{j \in [k]} c_j \right) \rightarrow \min. \tag{5}$$

The objective function O_2 is not linear due to the use of the max-function and hence must be linearized. This can be done using an auxiliary variable $\tilde{F} \in \mathbb{R}$, which is constrained by the maximum of the summed height and width of the SoC:

$$\tilde{F} \geq \sum_{i \in [l]} r_i, \tag{6}$$

$$\tilde{F} \geq \sum_{j \in [k]} c_j. \tag{7}$$

This relatively easy approach is possible because the linearized objective \tilde{C} function minimizes \tilde{F} :

$$\tilde{O} = \tilde{F} \rightarrow \min. \tag{8}$$

The issue of modeling Equation (4) remains, which is not linear. We propose both a linear approximation in Section 4.1, which is fast to calculate but does not yield an approximation error, and a nonlinear model in Section 4.2, which is slower than the linear approximation but has no error.

4.1. Linear Model

Since the area of a rectangle $F_{i,j}$ with edge length r_i and c_j cannot be calculated through means of a linear model, a linear approximation is required. The approach from Lacksonen et al. [22] for the factory layout problem can be applied here as well. Equation (4) is depicted in Figure 4: The iso-area-hyperbola $r_i c_j = F_{i,j}$ is shown in red in the space of row-heights r_i and column-widths c_j . Linearization of the iso-area-hyperbola is possible by introduction of an additional constraint for the aspect ratio of each tile. The aspect ratio $\eta \in (0, 1)$ limits the height and width of tiles for all $i \in [l]$ and $j \in [k]$: $r_i \geq c_j \eta$ and $r_i \leq c_j \eta^{-1}$. The constraint aspect ratio is shown in Figure 4a in blue.

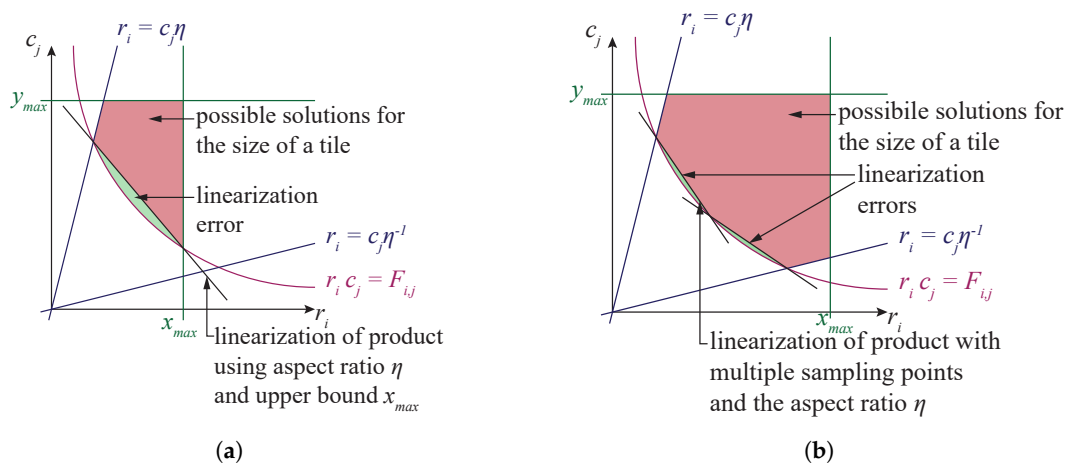


Figure 4. Area linearization [3]. (a) Simple approximation with single linear equation. (b) Reduced error through multiple linear approximations.

Figure 4a also shows the solution space as the red-shaded area. Following Equation (4), the area $r_i c_i$ of a tile i, j must be larger than its core with size $F_{i,j}$, i.e., $F_{i,j} \leq r_i c_i$. The iso-area-hyperbola is the lower left bound for the solution space. The maximum edge length of the tile further limits the solution space, given by the constraints: $c_j \leq y_{max}$ and $r_i \leq x_{max}$. Finally, the solution space is limited by the line equations for the aspect ratios η from Equations (10) and (11).

The iso-area-hyperbola is approximated by a line equation given by the intersections between the lines for the aspect ratios and the maximum edge length. This line equation is shown in black in Figure 4a. The resulting linearization error is plotted in green in Figure 4a. In general, it is possible to reduce this error by using multiple equally-spaced knots as shown in Figure 4b. Each linear equation connecting two adjacent knots intersected with the iso-area-hyperbola ($r_i c_j = F_{i,j}$) is called a 1-spline. While more 1-splines reduce the error, they also significantly increase the model complexity. Integer inequalities are required to determine in which spline a given solution is located. There are at least three additional integer inequalities per supporting point. Naturally, this reduces runtime performance.

To summarize, the linear optimization minimizes

$$\tilde{C} = \tilde{F} \longrightarrow \min, \quad (9)$$

subject to the following constraints with aspect ratio $\eta \in (0, 1)$:

$$r_i \geq \eta c_j \quad \forall i \in [l], \forall j \in [k], \quad (10)$$

$$c_j \geq \eta r_i \quad \forall i \in [l], \forall j \in [k], \quad (11)$$

$$r_i + c_j \geq \sqrt{F_{i,j}\eta} + \sqrt{F_{i,j}/\eta} \quad \forall i \in [l], \forall j \in [k]. \quad (12)$$

Equation (4) is approximated by Equation (12) for one single 1-spline as in Figure 4a. It can easily be deduced from the intersections of the iso-area-hyperbola and Equations (10) and (11). The required values for $\sqrt{F_{i,j}\eta} + \sqrt{F_{i,j}/\eta}$ can be precalculated before starting the optimization and thus are constants within the linear model.

4.2. Nonlinear Model

To remove the linearization error, SDPs can be used because they can express the red iso-area-hyperbola in Figure 4. We set kl variables $X_{k(i-1)+j}$ such that

$$X_{k(i-1)+j} = \begin{bmatrix} r_i & \sqrt{F_{i,j}} \\ \sqrt{F_{i,j}} & c_j \end{bmatrix} \succeq 0, \quad \forall i \in [l], \forall j \in [k]. \quad (13)$$

These matrices are premised to be positive semidefinite (i.e., “ $\succeq 0$ ”); thus, each principal minor is greater or equal to 0:

$$\det(X_{k(i-1)+j}) \geq 0, \quad (14)$$

$$\Leftrightarrow r_i c_j - F_{i,j} \geq 0, \quad (15)$$

$$\Leftrightarrow r_i c_j \geq F_{i,j}, \quad \forall i \in [l], \forall j \in [k]. \quad (16)$$

We formulate a SDP. The objective function minimizes the linearized variable $x \geq \max\{\sum r_i, \sum c_i\}$ using Equations (6) and (7):

$$x = \tilde{F} \longrightarrow \min, \quad (17)$$

subject to the following constraints.

We assign the corresponding area values to each matrix using the Frobenius inner product:

$$2\sqrt{F_{ij}} \leq \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_{k(i-1)+j} \right\rangle \leq 2\sqrt{F_{ij}}, \quad \forall i \in [l], \forall j \in [k]. \quad (18)$$

For each $i \in [l]$, the upper left entry of the matrices $X_{k(i-1)+j}$ has the same value for all $j \in [k]$ (this models r_i):

$$0 \leq \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, X_{k(i-1)+1} \right\rangle + \left\langle \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, X_{k(i-1)+j} \right\rangle \leq 0. \quad (19)$$

For each $j \in [k]$, the lower right entry of the matrices $X_{k(i-1)+j}$ has the same value for all $i \in [l]$ (this models c_j):

$$0 \leq \left\langle \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, X_j \right\rangle + \left\langle \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, X_{k(i-1)+j} \right\rangle \leq 0. \quad (20)$$

We model the maximum variable x for the objective function (this models $x \geq \sum r_i$ and $x \geq \sum c_j$):

$$0 \leq x + \sum_{i=1}^l \left\langle \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, X_{k(i-1)+1} \right\rangle, \quad (21)$$

$$0 \leq x + \sum_{j=1}^k \left\langle \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, X_j \right\rangle. \quad (22)$$

Again, areas of tiles are constrained by an aspect ratio η . Note that this aspect ratio is not violated by the relation between r_i and c_j . Rather, a component can find a rectangle inside the bounding box given by $r_i c_j$. This rectangle has the size of the core. The aspect ratio of its edges is greater than η . We formulate for all $i \in [l]$ and for all $j \in [k]$:

$$\sqrt{\eta F_{i,j}} \leq \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, X_{k(i-1)+1} \right\rangle, \quad (23)$$

$$\sqrt{\eta F_{i,j}} \leq \left\langle \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, X_j \right\rangle. \quad (24)$$

5. Results

5.1. Simulated Annealing (SA) vs. Particle Swarm Optimization (PSO)

Our approach is compared against [13], which uses PSO to map an application on a partially vertically-connected 3D mesh NoC with cores of different sizes. It is one of the most recent works on mapping in NoCs at the time of writing this paper, and it does account for cores of different areas, but it does not optimize area. To compare against this work, we use our cost function with the SA to map video object plane detection (VOPD) benchmark to a 3D-connected $4 \times 2 \times 2$ NoC and double video object plane detection (DVOPD) benchmark to a $4 \times 4 \times 2$ NoC with a varying number of vertical connections. The benchmark application graphs are from [23]. The other benchmarks from [23] are smaller and thus a comparison is not useful because both the PSO and the proposed heuristic algorithm using a simulated annealing will find the global minimum in a small design space in a short time. We chose an arbitrary but identical initial mapping for both benchmarks and both algorithms. We use 20 reruns for both PSO and simulated annealing so that both the algorithms have approximately the same computation time budget. The parameters of the PSO are given by [13] ($k1 = 1$, $k2 = 0.04$, $k3 = 0.02$). The parameters for the simulated annealing are: initial temperature 30, cooling

0.97, 1000 iterations. Both [13] and the our approach use the same objective function that minimizing bandwidth *times* communication hop distance. We disregard area because it is not used in [13] and therefore would skew the comparison. We change the TSV count in the NoCs to vary the mapping difficulty. The results are shown in Table 1 for VOPD and in Table 2 for DVOPD. The proposed heuristic algorithm allows for up to 15% improved performance with 2.564–3.125% better performance in average.

Table 1. Comparison of simulated annealing (SA) vs. particle swarm optimization (PSO) from [13] for VOPD benchmark. Network performance comparison (hop distance [HD] \times bandwidth [Mb]). Twenty reruns for PSO and simulated annealing with the same computational time budget for different TSV counts.

Vertical Connection Count	Hop Distance \times Bandwidth [Hd Mb]				Difference
	PSO		Proposed		
	mean	std	mean	std	
1	12,229	0	12,229	0	0%
2	10,591	581	9005	0	15%
3	8894	102	8659	0	3%
4	9013	364	8595	0	5%
5	8725	155	8595	0	1%
6	8723	148	8595	0	1%
7	8595	0	8595	0	0%
8	8595	0	8595	0	0%
Average Improvement					3.125%

Table 2. Comparison of SA vs. PSO from [13] for DVOPD benchmarks. Network performance comparison (hop distance [HD] \times bandwidth [Mb]). Twenty reruns for PSO and simulated annealing with the same computational time budget.

Vertical Connection Count	Hop Distance \times Bandwidth [Hd Mb]				Difference
	PSO		Proposed		
	mean	std	mean	std	
1	43,330	0	43,330	0	0%
2	38,274	163	37,954	395	1%
3	34,636	0	33,854	0	2%
4	34,217	674	32,382	0	5%
5	33,249	555	31,014	0	7%
6	32,351	699	30,168	0	7%
7	31,920	575	29,916	0	6%
8	30,767	679	29,744	0	3%
9	30,767	679	29,744	0	3%
10	30,318	453	29,712	0	2%
11	30,235	409	29,712	0	2%
12	29,764	69	29,712	0	0%
13	29,996	340	29,712	0	1%
14	29,805	208	29,712	0	0%
15	29,712	0	29,712	0	0%
16	29,712	0	29,712	0	0%
Average Improvement					2.563%

5.2. Linear vs. Non-Linear Model

We compare our linear and nonlinear models by generating results for the same inputs with both the LP and the SDP. We implement our models in MATLAB R2018a and they are available from

Github [24]. The LPs use IBM CPLEX 12.8.0 [25] as optimization engine. The SDPs use Mosek 8.1 [26]. We generate three random input benchmarks as in [3]. Iso-area cores are used for a fair comparison against conventional approaches:

1. A 3D SoC with two layers and five tiles, of which three tiles are in layer 1 and two tiles are in layer 2.
2. A 3D SoC with four layers and 10 tiles per layer connected by a 2×5 mesh NoC.
3. A 3D SoC with four layers and 20 tiles per layer connected by a 4×5 mesh NoC.

Cores are set to be 10 mm^2 large. Routers with five ports require 1 mm^2 . The router area is linearly proportional to port count depending on the position of the router in the network. TSV arrays, which vertically connect routers, are 2 mm^2 large. The aspect ratio is limited by $\eta = 0.1$. We run the optimization 50 times to average runtime on an Intel Core i7-6700 (eight cores at 3.4 GHz) using Windows 10.

The results for performance, runtime and model properties are reported in Table 3:

- **Performance.** In benchmark 1, the summed chip area is 68.7 mm^2 from the LP and 59.8 mm^2 from the SDP. In benchmark 2, the summed chip area is 832 mm^2 from the LP and 695 mm^2 from the SDP. In benchmark 3, the summed chip area is 1272 mm^2 from the LP and 1188 mm^2 from the SDP. Since in the lowest layer there is no TSV area required (there are no keep-out-zones using via-middle-process-flow), this layer is smaller.
- **Runtime.** The difference in runtime between LP and SDP is between $6\times$ and 31%. The LP loses its runtime advantage for larger inputs.
- **Model Properties.** We also report inequality and variable count. The linear model requires $2kl + 2$ inequalities and $k + l + 1$ variables. The nonlinear model requires $(kl)^2 + k + l + 2$ inequalities and $kl + 1$ variables. Thus, the SDP has considerably more variables and inequalities. However, both models are very small in comparison to common use cases for LP and SDP solvers with millions of variables and equations. Therefore, the models do not largely differentiate in terms of memory usage.

Table 3. Area, runtime, inequality count and variable count comparison between linear and nonlinear model (runtime average of 50 reruns) [3].

Layer	Area [mm^2]								
	5 PEs			40 PEs			80 PEs		
	LP	SDP	Δ	LP	SDP	Δ	LP	SDP	Δ
1	43.0	36.8	−14.4%	211	178	−15.6%	364	301	−17.4%
2	25.7	23.0	−10.5%	222	180	−18.9%	379	313	−17.4%
3	—	—	—	214	183	−14.5%	378	313	−17.2%
4	—	—	—	185	154	−16.8%	316	261	−17.4%
Average Area Reduction			−12.5%	−16.5%			−17.3%		
Average runtime [s]									
	0.4	2.9	+625%	3.9	7.5	+92.3%	12.2	16.0	+31.1%
Inequality count									
	16	31	+94%	88	436	+395%	168	1644	+879%
Variable count									
	9	21	+133%	32	112	+250%	40	200	+400%

5.3. SA for Area-Aware Core Mapping with Linear vs. Nonlinear Models

As introduced in the related work, Ref. [2] compares against our approach as it conducts core mapping and accounts for varying area of cores. Only quadratic-shaped cores are mapped to a 2D mesh NoC in that work. The reference's objective function does not target a low area but minimizes transmission energy. We compare the results from our nonlinear model with results from linear models from [2] using the three multimedia benchmarks provided, which cover video and audio decoder and encoder. The data streams for the benchmarks are taken from [23] and the cores' area from [2].

The results for area, hop distance \times bandwidth and total bandwidth are reported in Table 4. The three figures are measured following Def. 6. For our experiments, we take the mapping from [23] as baseline. Next, we optimize this mapping with the SDP to demonstrate the advantages of nonlinear models at a realistic benchmark. Then, we generate an area-efficient initial solution following the greedy algorithm introduced in Section 3.2. Then, we conduct two separate experiments with our SA cost function. First, we set the weight for area in our cost function (Equation (2)) w_1 to zero, so that only communication is minimized. This resembles the behavior of the objective function in [2]. Second, the weights in the objective function (Equation (2)) are normalized such that area and communication are accounted for simultaneously. The simulated annealing is executed 20 times with 15,000 iterations, an initial temperature of 30 and a cooling of 0.98. The aspect ratio is limited by $\eta = 0.1$ in both these experiments. The results of all runs are averaged and the standard deviation is calculated. A single run of the SA terminates after 17 minutes on a Windows 10 workstation using an Intel i7-7740X processor (8 cores at 4.3 GHz).

Table 4. Area and network performance comparison of mapping to a 2D-mesh NoC using the simulated annealing (SA) from [2]. The SA is executed with 20 reruns, an initial temperature of 30, cooling of 0.98 and 15,000 iterations. The aspect ratio is limited by $\eta = 0.1$ [3].

			Area [mm ²]			Hop Distance \times Bandwidth			Bandwidth [Bits]		
			mean	std	Ratio	mean	std	Ratio	mean	std	Ratio
H256 dec	mp3 dec	Baseline [2]	11,301	—	—	19,858	—	—	4060	—	—
		Baseline with SDP	10,178	—	−9.94%	19,858	—	0.0%	4060	—	0.0%
		Initial solution	7902	—	−30.1%	33,707	—	+69.7%	7994	—	+96.9%
		SA communication ($w_1=0$)	11,699	1598	+3.52%	20,449	404	+2.98%	4265	201	+5.05%
		Normalized SA with SDP	8244	505	−27.1%	21,280	624	+7.16%	4452	674	+9.66%
H263 enc	mp3 dec	Baseline [2]	12,535	—	—	255,324	—	—	84,884	—	—
		Baseline with SDP	10,178	—	−18.8%	255,324	—	0.0%	84,884	—	0.0%
		Initial solution	6993	—	−44.2%	525,537	—	+106%	85,244	—	+0.42%
		SA communication ($w_1=0$)	15,762	1723	−25.7%	241,479	15,333	−5.42%	73,012	14,302	−14.0%
		Normalized SA with SDP	10,474	2148	−16.4%	250,187	14,763	−2.0%	73,161	17,497	−13.8%
mp3 enc	mp3 dec	Baseline [2]	8568	—	—	17,546	—	—	4085	—	—
		Baseline with SDP	8091	—	−5.57%	17,546	—	0.0%	4085	—	0.0%
		Initial solution	7281	—	−15.0%	39,171	—	+123.3%	6560	—	+60.1%
		SA communication ($w_1=0$)	10,779	1460	+25.8%	17,341	342	−1.17%	5065	906	+24.0%
		Normalized SA with SDP	8516	796	−0.61%	17,572	487	+0.15%	4974	902	+21.8%

In the first experiment, the heuristic from [2] and the proposed cost function with SA produce similar results: hop distance \times bandwidth with between 5% better and 3% worse, while total bandwidth shows more variation with 14% better to 24% worse. Since we do not optimize area, the results are purely by chance. For the second experiment, our proposed algorithm shows the following improvement: In the best case, the H263 enc mp3 dec, the proposed cost function for a simulated annealing improves area by 16.4%, hop distance \times bandwidth by 2% and total bandwidth by 13.8%. Both experiments show that the quality of results depends on the structure of the input data. While the H263 enc mp3 dec benchmark offers large optimization potential, the mp3 enc mp3 dec benchmark shows minor differences in area and hop distance \times bandwidth. The H256 dec mp3 dec benchmark offers the large area improvement by 27%. However, this comes at a price of higher communication costs by up to 9.66%.

6. Conclusions

To summarize our paper, we showed a cost function for a simulated annealing algorithm to optimize area and communication during core mapping for NoC-based SoCs. As a novel feature, the heuristic is area-aware, which is a new requirement from heterogeneous core and IP areas in modern SoCs. We conducted experiments to compare the proposed algorithm against the state-of-the-art in the subject areas. First, we justify the use of simulated annealing over other heuristic searches by comparison against a recent work on core mapping using PSO. SA performs 2.5–3% better on average for different benchmarks with the same computational time budget. Second, we compare linear with nonlinear models to optimize area within the simulated annealing. We find that the nonlinear SDP yields 12.5–17.3% reduced area for different randomly generated inputs, which is within the expectations for the chosen linearization (Ref. [22] reports 20% area error for similar problems). In addition, as expected, the runtime of the SDP is longer than of a linear model. However, even for the largest example, the SDP only runs 3.6 seconds longer absolutely. This is a rather small price to pay for 17.3% better area. Third, our cost function is compared against the state-of-the-art mapping including area. For a H263 enc mp3 dec benchmark, our approach generates 16.4% better area, 2% better hop distance \times bandwidth and 13.8% better total bandwidth. These three experimental setups show that our approach is practical, as it reduces area and communication costs for real-world based benchmarks, efficient, as it has runtime as state-of-the-art, and effective, as it allows for reduced area by elimination of linearization errors. By that, we demonstrate the practical applicability of nonlinear models in EDA.

Author Contributions: Data curation, D.E.; Project administration, T.P.; Supervision, A.G.-O.; Validation, L.B.; Writing—original draft, J.M.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the German Research Foundation (DFG) project PI 447/8 and GA 763/7. This work was supported by a fellowship within the Internationale Forschungsaufenthalte für Informatikerinnen und Informatiker (IFI) programme of the German Academic Exchange Service (DAAD).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Murali, S.; Coenen, M.; Radulescu, A.; Goossens, K.; De Micheli, G. A Methodology for Mapping Multiple Use-Cases onto Networks on Chips. In Proceedings of the Conference on Design, Automation and Test in Europe, Munich, Germany, 6–10 March 2006. [\[CrossRef\]](#)
2. Srinivasan, K.; Chatha, K.S.; Konjevod, G. Linear-programming-based techniques for synthesis of network-on-chip architectures. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2006**, *14*, 407–420. [\[CrossRef\]](#)
3. Joseph, J.M.; Ermel, D.; Drewes, T.; Bamberg, L.; García-Oritz, A.; Pionteck, T. Area Optimization with Non-Linear Models in Core Mapping for System-on-Chips. In Proceedings of the 8th International Conference on Modern Circuits and Systems Technologies (MOCAS), Thessaloniki, Greece, 13–15 May 2019. [\[CrossRef\]](#)
4. Lin, L.-Y.; Wang, C.-Y.; Huang, P.-J.; Chou, C.-C.; Jou, J.-Y. Communication-driven task binding for multiprocessor with latency insensitive network-on-chip. In Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), Shanghai, China, 21–21 January 2005. [\[CrossRef\]](#)
5. Satish, N.; Ravindran, K.; Keutzer, K. A Decomposition-based Constraint Optimization Approach for Statically Scheduling Task Graphs with Communication Delays to Multiprocessors. In Proceedings of the Conference on Design, Automation and Test in Europe, Nice, France, 16–20 April 2007.
6. Zarándy, Á. *Focal-Plane Sensor-Processor Chips*; Springer: Berlin, Germany, 2011.
7. Rhee, C.-E.; Jeong, H.-Y.; Ha, S. Many-to-many core-switch mapping in 2-D mesh NoC architectures. In Proceedings of IEEE International Conference on Computer Design (ICCD): VLSI in Computers and Processors, San Jose, CA, USA, 11–13 October 2004. [\[CrossRef\]](#)
8. Murali, S.; Benini, L.; De Micheli, G.; De Micheli, G.; De Micheli, G. Mapping and Physical Planning of Networks-on-chip Architectures with Quality-of-service Guarantees. In Proceedings of the 2005 Asia and South Pacific Design Automation Conference, Shanghai, China, 18–21 January 2005. [\[CrossRef\]](#)

9. Ostler, C.; Chatha, K.S. An ILP Formulation for System-level Application Mapping on Network Processor Architectures. In Proceedings of the Conference on Design, Automation and Test in Europe, Nice, France, 16–20 April 2007.
10. Ozturk, O.; Kandemir, M.; Son, S.W. An ilp based approach to reducing energy consumption in nocbased CMPS. In Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED), Portland, OR, USA, 27–29 August 2007. [[CrossRef](#)]
11. Hu, J.; Marculescu, R. Energy-aware mapping for tile-based NoC architectures under performance constraints. In Proceedings of the 2003 Asia and South Pacific Design Automation Conference, Kitakyushu, Japan, 21–24 January 2003.
12. Tang Lei.; Kumar, S. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In Proceedings of the Euromicro Symposium on Digital System Design, Belek-Antalya, Turkey, 1–6 September 2003. [[CrossRef](#)]
13. Manna, K.; Swami, S.; Chattopadhyay, S.; Sengupta, I. Integrated Through-Silicon Via Placement and Application Mapping for 3D Mesh-Based NoC Design. *ACM Trans. Embedded Comput. Syst.* **2016**, *16*. [[CrossRef](#)]
14. Cong, J.; Wei, J.; Zhang, Y. A thermal-driven floorplanning algorithm for 3D ICs. In Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD), San Jose, CA, USA, 7–11 November 2004. [[CrossRef](#)]
15. Joseph, J.M.; Ermel, D.; Bamberg, L.; García-Ortiz, A.; Pionteck, T. System-level optimization of Network-on-Chips for heterogeneous 3D System-on-Chips. *arXiv* **2019**, arXiv:cs.AR/1909.13807.
16. Lu, Z.; Xia, L.; Jantsch, A. Cluster-based Simulated Annealing for Mapping Cores onto 2D Mesh Networks on Chip. In Proceedings of the 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, Bratislava, Slovakia, 16–18 April 2008. [[CrossRef](#)]
17. Zhong, L.; Sheng, J.; Jing, M.; Yu, Z.; Zeng, X.; Zhou, D. An optimized mapping algorithm based on Simulated Annealing for regular NoC architecture. In Proceedings of the 9th IEEE International Conference on ASIC, Xiamen, China, 25–28 October 2011. [[CrossRef](#)]
18. Kashi, S.; Patooghy, A.; Rahmatiy, D.; Fazeli, M.; Kinsy, M.A. Application Specific Networks-on-Chip Synthesis: An Energy Efficient Approach. In Proceedings of the 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Hong Kong, China, 8–11 July 2018. [[CrossRef](#)]
19. Li, B.; Wang, X.; Singh, A.K.; Mak, T. On runtime communication- and thermal-aware application mapping in 3D NoC. In Proceedings of the 11th IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Seoul, Korea, 19–20 October 2017.
20. Murali, S.; Micheli, G.D. Bandwidth-constrained mapping of cores onto NoC architectures. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, Paris, France, 16–20 February 2004. [[CrossRef](#)]
21. Korte, B.; Vygen, J. *Combinatorial Optimization: Theory and Algorithms*, 5th ed.; Springer: Berlin, Germany, 2012.
22. Lacksonen, T.A. Static and Dynamic Layout Problems with Varying Areas. *J. Oper. Res. Soc.* **1994**, *45*, 59–69. [[CrossRef](#)]
23. Sahu, P.K.; Chattopadhyay, S. A survey on application mapping strategies for Network-on-Chip design. *J. Syst. Archit.* **2013**, *59*, 60–76. [[CrossRef](#)]
24. Joseph, J. System-level Optimization of NoCs for Heterogeneous 3D SoCs. 2019. Available online: <https://github.com/jmjos/A-3D-NoC-DSE> (accessed on 8 October 2019).
25. IBM. *ILOG CPLEX Optimization Studio CPLEX User's Manual 12.8*; Armonk: New York, NY, USA, 2017.
26. Mosek ApS. *Mosek User Manual*; Mosek ApS: Copenhagen, Denmark, 2018.

