J Supercomput (2015) 71:995–1017
DOI 10.1007/s11227-014-1348-x

Application mapping algorithms for mesh-based network-on-chip architectures

Suleyman Tosun · Ozcan Ozturk · Erencan Ozkan · Meltem Ozen

Published online: 22 November 2014

Abstract Due to shrinking technology sizes, more and more processing elements and memory blocks are being integrated on a single die. However, traditional communication infrastructures (e.g., bus or point-to-point) cannot handle the synchronization problems of these large systems. Using network-on-chip (NoC) is a step towards solving this communication problem. Energy- and communication-efficient application mapping is a previously studied problem for mesh-based NoC architectures; however, there is still need for intelligent mapping algorithms since current algorithms either take too much running time or do not determine accurate results. To fill this need, in this study, we propose two mapping algorithms (one based on simulated annealing and one based on genetic algorithm) for energy- and communication-aware mapping problems of mesh-based NoC architectures. We compare these two algorithms with an integer linear programming-based method and a heuristic method using several multimedia and synthetic benchmarks.

Keywords NoC · Mesh topology · Mapping · Genetic algorithm · Simulated annealing · Integer linear programming · Heuristics

S. Tosun (✉)
Computer Engineering Department, Hacettepe University, Beytepe, Ankara, Turkey
e-mail: stosun@hacettepe.edu.tr; stosun@gmail.com

O. Ozturk
Computer Engineering Department, Bilkent University, Bilkent, Ankara, Turkey
e-mail: ozturk@cs.bilkent.edu.tr

E. Ozkan, M. Ozen
Computer Engineering Department, Ankara University, Golbasi, Ankara, Turkey
e-mail: erencanozkan@gmail.com

M. Ozen
e-mail: meltemmozen@gmail.com

1 Introduction

Network-on-chip (NoC) has become the central communication paradigm for system-on-chip (SoC) designs after it was introduced over a decade ago [1,2]. This method provides a suitable and economical solution to integrate complex systems on a single chip for shrinking very-large-scale integration (VLSI) technology sizes and make it possible to synchronize these large numbers of components [3,4]. There have been some earlier work that benefit such a scalable network topology and various scalable applications as well [5–8]. NoC architectures consist of several interconnected heterogeneous blocks called intellectual property (IP) blocks ranging from general or special-purpose processors to embedded memories, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), etc. Communication among these IP blocks is carried out by sending packets via network components (e.g., network adapters, routing nodes, and links). Although NoC designs ease the communication of IP blocks and increase the system throughput tremendously, designing the overall system includes several trade-offs in topology selection, routing strategy selection, application mapping, and scheduling on network tiles. Each of these steps has an effect on the overall energy consumption, one of the important criteria in system design. NoC architectures can be constructed using regular topologies or irregular (custom) topologies. Both topology types exhibit advantages and disadvantages: Irregular topologies suit the optimization of different requirements such as link size and number of routers to be used; however, they are not reusable. On the other hand, regular topologies can be reused and are easy to design. Most multi-core architectures employ regular topologies, especially mesh topology.

Energy-aware application mapping is a relevant problem for mesh-based NoC designs. For an application with n tasks, we can produce $n!$ solutions. This huge solution space makes exhaustive search algorithms infeasible for large problems. Nonetheless, several optimization techniques have been developed to improve design parameter values of the stated problems. For example, in our previous study, we presented an integer linear programming (ILP)-based method that obtains optimal mappings [9]. However, ILP-based method takes too much CPU runtime when the input size n is high. To remedy this runtime problem, we previously proposed a heuristic method called CastNet [10]. The problem with the heuristic method is that it does not guarantee the optimal solutions. In fact, it may obtain unacceptable solutions that are too far from the optimal one since heuristic methods may get stuck at local minima.

Our ILP- and heuristic-based methods are two far ends to the stated problem when the concern is accuracy and CPU runtime together. Motivated by this observation, we developed two metaheuristic methods that take less computation time and do not stuck at local minima. In this paper, we propose these two optimization techniques for energy-aware application mapping for NoCs; one technique is based on a genetic algorithm (GA) and the other uses simulated annealing (SA). We extensively compare and contrast these algorithms through several multimedia benchmarks with our ILP-based and heuristic methods. We also evaluate these algorithms with synthetic benchmarks through randomly generated task graphs. We show that our GA- and SA-based methods generate very promising results in execution time and energy consumption.

The paper is organized as follows: We present related work in Sect. 2. We present the problem definition and the energy model in Sect. 3. In Sect. 4, we explain the mapping algorithms. We give the experimental results in Sect. 5. Finally, we conclude the paper in Sect. 6.

2 Related work

While several studies have explored better solutions to the mapping problem from different angles, in this paper, our focus is on algorithms that target energy minimization.

Murali and De Micheli [11] proposed a heuristic algorithm called NMAP, which maps the given application onto a 2D mesh under bandwidth constraints targeted to minimize the energy consumption of the final design. In their algorithm, they initially map the tasks based on their communication weights. They then calculate the communication cost of this mapping using Dijkstra's shortest-path algorithm. Finally, they refine the mapping by iteratively improving it by swapping the pairs of nodes.

Another heuristic method (called ONYX) is presented in [12]. In this algorithm, the authors propose a method that selects the tasks to be mapped based on the assigned priority. Then the algorithm maps the selected task by searching the candidate tiles on a lozenge-shaped path. This is extended in CastNet [10] using the symmetry of the mesh topologies to select the initial tiles. We used CastNet in our evaluation as the heuristic method.

In [13], the authors proposed a binomial IP mapping technique, called BMAP. This algorithm first maps the given algorithm on a 1D mesh. Then, it iteratively applies a binomial merge, which is a simple tree merging problem, until the target mesh topology is achieved.

Hu and Marculescu [14] proposed an energy-aware mapping algorithm based on branch-and-bound method. This algorithm walks through the searching tree that represents the solution space. The authors compare their algorithms with a SA-based method and show up to 51.7 % energy savings.

Genetic algorithm and simulated annealing are commonly used metaheuristics for NP-hard optimization problems. For example, SA has been used for cluster-based mapping cores onto 2D mesh NoCs [15]. GA has also been used for the mapping problem aiming different objective criteria [16–21]. In [16], the authors proposed a chaos-genetic algorithm to minimize energy consumption of the design; however, the algorithm does not bring much improvement over earlier ones. Ascia et al. [17] proposed a GA-based method that maps the application onto the mesh to optimize multi-objective functions. Hung et al. [18] also proposed a GA-based method to reduce the communication cost of thermal balanced designs.

While we aim to map the given application on a given environment to minimize the energy consumption in this study, there have been prior efforts [19–21] that apply multi-objective evolutionary optimization algorithms aiming to minimize the chip area, the power consumption, and the execution time of the application. Since the mapping platform of our work is different than these studies and we do not consider the area constraints of the design, we did not compare our GA- and SA-based algorithms

with them. The reason we did not consider area constraint is that the chips of the current technology are abundant of transistors.

There have been ILP-based methods for the solution of mapping problems. Srinivasan and Chatha [22] presented an ILP formulation for mapping tasks to mesh architectures, comparing it with [11], and developed a better design. In our earlier work, we also formulated the mapping problem and used the 0–1 ILP for our formulations [9,23], in which variables are restricted to being either 0 or 1. In this paper, we restate the formulations to make this paper self-contained. ILP seems the best option for linear optimization problems if the number of variables is limited; however, when it is applied to the problems with huge number of variables, it takes too much computation time to obtain optimal solutions.

There is a similar comparison work presented in an earlier study [24]. Our work differs from this study in three aspects. First, we propose our own methods for each optimization technique. Second, we use a different heuristic algorithm, CastNet [10], to compare with other optimization methods. Finally, we compare our methods using real benchmarks rather than only on random graphs.

3 Problem definition and energy model

In this section, we define the application mapping problem, illustrate the input application, and describe the target architecture.

3.1 Energy model

In this study, our goal is not to minimize the energy consumed by the functional blocks because that is independent of the generated topology; rather, we try to minimize the energy consumed by the network resources. The energy consumption of the network is directly proportional to the amount of bit transitions on the network. To estimate the energy consumption of the NoC architecture, we should use an energy model based on the total bit transitions. This work uses the well-accepted energy model given in [14] to calculate the total communication energy $E_{T_{\text{bit}}}$ as

$$E_{T_{\text{bit}}} = E_{S_{\text{bit}}} + E_{B_{\text{bit}}} + E_{W_{\text{bit}}} + E_{L_{\text{bit}}}, \tag{1}$$

where $E_{S_{\text{bit}}}$, $E_{B_{\text{bit}}}$, $E_{W_{\text{bit}}}$, and $E_{L_{\text{bit}}}$ represent the energy consumed by the switch, the buffer, interconnection wires inside the fabric, and the links, respectively. Since the energy consumption of the buffering $E_{B_{\text{bit}}}$ and the internal wires $E_{W_{\text{bit}}}$ are negligible [14], Eq. (1) can be reduced to:

$$E_{T_{\text{bit}}} = E_{S_{\text{bit}}} + E_{L_{\text{bit}}}. \tag{2}$$

Then, the average energy consumption of sending one bit of data from core v_i to core v_j can be calculated by the formula:

$$E_{T_{\text{bit}}}^{v_i,v_j} = \eta_{v_i,v_j} E_{S_{\text{bit}}} + (\eta_{v_i,v_j} - 1) E_{L_{\text{bit}}}, \tag{3}$$

where η_{v_i,v_j} is the number of routers the bit passes through. If a bit passes through η routers, clearly, it also passes through $\eta - 1$ links. Let v_a and v_b be the vertices mapped onto the tiles t_i and t_j, respectively. Communication between two tiles with a communication weight $w_{a,b}$ can be calculated by (4) as

$$E_{\text{total}}^{t_{i,j}} = w_{a,b} E_{T_{\text{bit}}}^{v_a,v_b}. \tag{4}$$

Finally, we can determine the total energy consumption of the NoC architecture as follows:

$$E_{\text{NoC}} = \sum_{\forall e_{i,j} \in E} E_{\text{total}}^{t_{i,j}}. \tag{5}$$

3.2 Problem definition

We use the well-accepted abstract graph models, which we call the weighted communication task graph (WCTG) and the topology graph (TG), to represent the input application and target architecture, respectively. We define WCTG and TG as follows:

Definition 1 A WCTG is a graph $G(V, E)$ where each vertex $v_i \in V$ represents a task (i.e., a node) in the application and each edge $e_{i,j} \in E$ represents a dependency between two tasks v_i and v_j. The amount of data transferred between v_i and v_j is represented by the weight $w_{i,j}$ for each $e_{i,j}$ in bits per second.

Definition 2 A TG is a graph (P, L), where each vertex $p_i \in P$ denotes a processing core (i.e., a tile) in the topology and each edge denotes a physical link $l_{i,j} \in L$ between p_i and p_j.

Figure 1a shows the WCTG of the MPEG-4 benchmark. In this WCTG, we have 12 vertices and 13 edges, where the weights of the edges represent the amount of data transferred between two tasks in Kbits/s. An example TG with 16 cores (i.e., a 4×4 mesh) is given in Fig. 1b, where mapping MPEG-4 nodes onto a 4×4 mesh NoC architecture is shown.

Using the above definitions, the application mapping problem can be formulated as follows:

Given a WCTG and a TG those satisfy

$$|V| \leq |P|, \tag{6}$$

find a one-to-one mapping function $F : V \rightarrow P$ from WCTG to TG that minimizes the energy consumption of the NoC.

We can state the mapping problem mathematically as:

$$\textbf{min} : \ \{E_{NoC}\} \tag{7}$$

such that:

$$\forall v_i \in V, \exists p_k \in P, F(v_i) = p_k \tag{8}$$

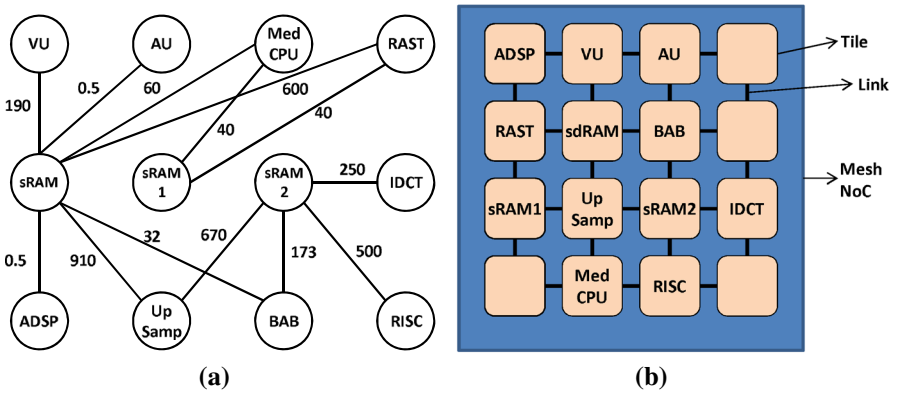


Fig. 1 **a** WCTG of MPEG-4 and **b** mapping MPEG-4 onto 4×4 mesh NoC

and

$$\forall v_i \neq v_j \in V, F(v_i) \neq F(v_j). \tag{9}$$

3.3 Routing scheme

The routing algorithm is an important factor in NoC designs because it establishes the path of the messages between cores. The routing algorithms for NoCs must be deadlock free (i.e., it must be guaranteed that packets will not block the network) to meet their predefined performance constraints. We can group routing algorithms, based on their path decision function, as deterministic or adaptive.

Deterministic routing algorithms always supply the same path between source and destination nodes. A well-known example of a dimension-order deterministic algorithm for 2D mesh networks is XY routing [25]. In mesh topologies, routers are identified by their coordinates (x, y). If a packet traveling from the current router $C(x, y)$ to the destination router $D(x, y)$, it is first routed horizontally (i.e., in x dimension) to east or west neighboring router until x coordinate of the packet's current router is aligned with destination router's x coordinate. The packet is then forwarded vertically (i.e., in y dimension) to south or north neighboring router after comparing y coordinates of current and destination routers until the packet arrives to its destination. The advantages of deterministic routing algorithms are twofold: Low implementation cost and traffic predictability. Routers used for deterministic routing algorithms require fewer resources than ones for adaptive algorithms. Additionally, deterministic routing algorithms offer a predictable traffic pattern, which makes it easier for a designer to analyze the network. On the other hand, adaptive routing algorithms have better resilience to deadlocks. They can be used in high-traffic networks with high traffic and in networks with limited bandwidth.

In this study, since we do not consider link bandwidth constraints in the mapping step, we use an XY routing algorithm to decrease the energy consumption of the final design. However, one could use an adaptive or partially adaptive routing algorithm to prevent deadlocks in the network. Since we compare mapping algorithms in this

study, we do not further elaborate on the details of the candidate routing mechanism; we simply use XY routing algorithms for all five mapping methods.

4 Mapping algorithms

In this section, we explain five mapping methods that use different optimization techniques with different runtime complexities. These methods are based on random, SA, GA, ILP, and heuristic optimization techniques.

4.1 Random mapping

In this method, we randomly map tasks onto cores of the mesh. We produce 1,000 solutions and select the one with the best energy savings. As one can see, there is no intelligent optimization technique in this method except to choose the better of two candidates. Additionally, each run may output different solutions. We implemented this method for comparison purposes.

4.2 SA-based method

Simulated annealing is a probabilistic method for finding the global optimum of a cost function [26]. It is modeled on the behavior of condensed matter at low temperatures. The annealing process starts with an initial configuration and continuously reduces the temperature of the system in search for a better configuration. If the process obtains a better energy values in a configuration, it directly accepts this configuration. Otherwise, it accepts the new solution based on acceptance probability function. In this way, it makes uphill moves from a local minimum in an attempt to jump to a valley where the global minimum might reside.

Simulated annealing is a good choice to solve the mapping problem stated in this work. We give the pseudocode of an SA-based application mapping algorithm in Algorithm 1, where we randomly map tasks onto the mesh (line 2) to obtain an initial mapping. We then calculate the total communication cost of the initial mapping using Eq. (10). One can see that we use the total communication cost of the design instead of the total energy consumption in the equation, even though our final goal is to minimize the energy consumption. This is because, as we stated in Sect. 3.1, the energy consumption of a NoC system is proportional to the amount of bit transitions in the network. Therefore, when we minimize the total bit transitions (i.e., the communication cost) of the system, the system's total energy consumption is proportionally minimized.

$$\text{Comm} = \sum_{\forall e_{i,j} \in E} (\eta_{f(v_i), f(v_j)} - 1) w_{i,j} \tag{10}$$

After the initial mapping is determined, we set temperature to its highest value. Temperature parameter in our mapping problem is analogous to Manhatten distance between two nodes on the mesh. If the distance between two exchange nodes is high, the temperature is high and vice versa. A previous study [24] states that the algorithm

Algorithm 1: SA-based mapping

```
Data: G, M
Result: T, E_NoC, Comm
1  begin
2      T = Random_Initial_Mapping(G, M)
3      C = Calculate_Comm(T)
4      T_best = T
5      C_best = C
6      Temperature = ⌈10 ln |P|⌉
7      for i → 0 to |P|^2 do
8          R = 0
9          while R < 10 do
10             T' = neighbor(T)
11             C' = Calculate_Comm(T')
12             ΔC = C − C'
13             Generate a random variable α, 0 ≤ α ≤ 1
14             if ΔC ≤ 0 or α ≤ e^(−ΔC)/Temperature then
15                 T = T'
16                 R = 0
17             else
18                 R++
19             if R = 0 ∧ C < C_best then
20                 T_best = T
21                 C_best = C
22         Decrement Temperature
23     T = T_best
24     E_NoC = Calculate_Energy(T)
25     Comm = C_best
26     return T, E_NoC, and Comm
```

obtains good results when the initial temperature is selected to be $\lceil 10 \ln |P| \rceil$, where $|P|$ is the number of tiles in the mesh.

After the temperature of the system is initialized, the algorithm executes two nested loops. While the external loop searches for global minima, the internal loop tries to refine the local solution. We limit the number of external loop iterations to $|P|^2$ as suggested in [24]. The internal loop randomly selects two nodes and swaps them to determine a new solution. It then evaluates if the new solution is better than the solution at hand. If it is, the solution is accepted to be the current solution. Otherwise, it generates a random variable α, where $0 \leq \alpha \leq 1$, and compares it with the acceptance probability function $e^{(-\Delta C)/\text{temperature}}$. If the result of the function is higher than α, the new move is accepted. At high temperatures, the acceptance probability is also high. When we lower the temperature of the system, the acceptance probability decreases. We limit the iteration of the internal loop to 10 consecutive rejects. After each iteration, we decrement temperature of the system and start a new iteration accepting the solution at hand as our initial solution for the new iteration.

In Fig. 2, we illustrate an example iteration of our SA-based mapping algorithm. Figure 2a shows an example WCTG. Figure 2b and c shows two mappings (before and

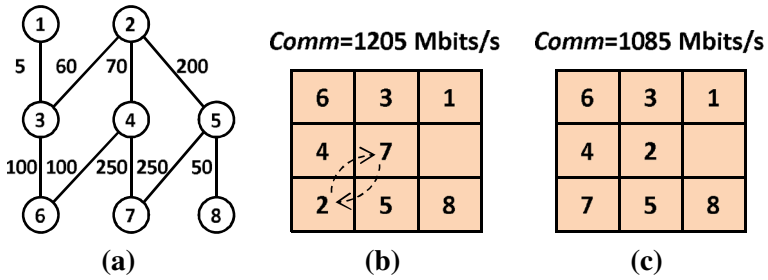


Fig. 2 An example iteration for SA-based method. **a** An example WCTG, **b** mapping before node swapping, **c** mapping after swapping two nodes

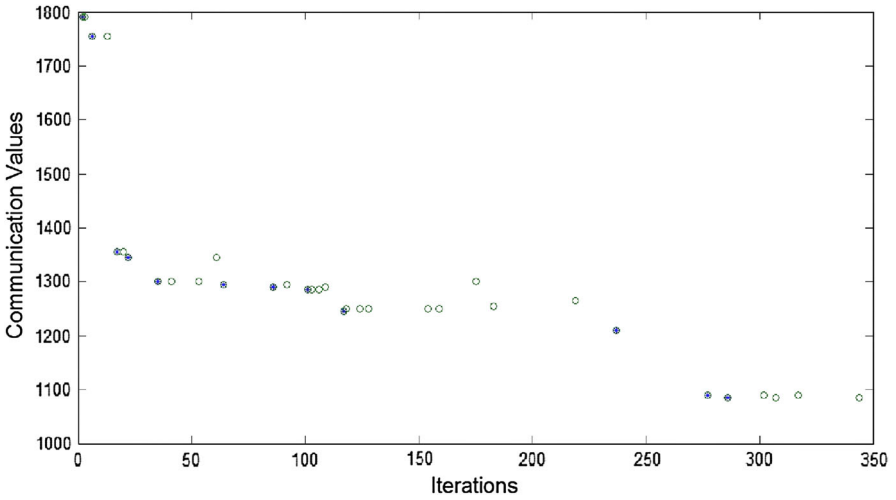


Fig. 3 An example iteration for accepted moves of SA-based algorithm

after swapping nodes v_2 and v_7, respectively). The communication cost of mapping before the swap is calculated as Comm = 1,205 Mbits/s. After swapping nodes v_2 and v_7, the communication cost becomes Comm = 1,085 Mbits/s, which is accepted as the new solution. The algorithm continues executing the swapping iteration until the predefined iteration value is reached.

In Fig. 3, we show all accepted solutions by our SA-based mapping tool for the WCTG given in Fig. 2a. In this graphics, the horizontal axis shows the iterations while the vertical axis shows the total communication cost. As can be seen from this graphics, while SA-based method keeps the best result during the run (given by * symbol in the graphics) it also makes uphill moves by selecting worse solutions. By doing this, it searches for a better minimum than the one at hand.

4.3 GA-based method

Genetic algorithm is another metaheuristic method for solving optimization and search problems [27]. Inspired by the biological process of evolution, it mimics the evolutionary functions of selection, mutation, and crossover processes to determine optimum

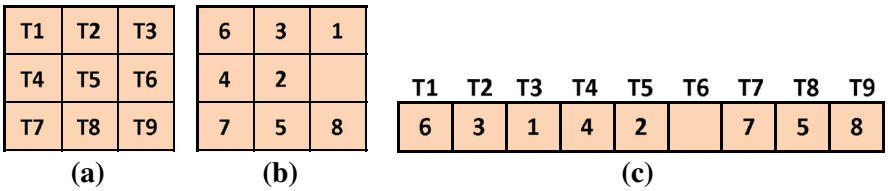


Fig. 4 **a** The sequence for a 3×3 mesh, **b** mapping of WCTG in Fig. 3a onto a 3×3-mesh, **c** chromosome representation of the mapping

or approximate solutions. In nature, these biological functions are repeated over generations and result in individuals best fitted to their environment. The GA has been shown to be very attractive for computationally intense multi-parameter optimization problems; thus, we also employ it for our mapping problem.

In our GA method, we first create a chromosome structure that represents a valid mapping. In this representation, a chromosome is constructed by a string of genes, each of which represents a tile of mesh. In short, gene i represents tile i in a chromosome. Therefore, the size of a chromosome is limited to the number of tiles in our mesh. We start our GA implementation with a set of randomly generated chromosomes, which forms our initial population. Figure 4 demonstrates the relation between a mapping and the corresponding chromosome. In Fig. 4a, we give the tile sequence for a 3×3 mesh, where the sequence starts from the tile in the upper left corner of the mesh and ends on the tile in the bottom right corner. Figure 4b shows a random mapping of WCTG in Fig. 4a onto a 3×3-mesh NoC architecture. The chromosome representation of this mapping is given in Fig. 4c. After creating n individuals for our population, we calculate the fitness of each chromosome, which is the total communication cost (i.e., Comm), by Eq. (10).

The optimization process starts from the randomly created initial population and repetitively applies crossover, mutation, and selection operators to generate new individuals. In Fig. 5, we demonstrate how the crossover and mutation operations are applied on randomly selected chromosome pairs. In Fig. 5a, we apply the crossover operation on two individuals in our population. In this operator, we randomly determine a cut point on the chromosome. We then swap the second portions of the chromosomes with each other. After swapping, we obtain two child individuals. However, the newly generated chromosomes are generally invalid because they do not meet the constraints given in Sect. 3.2. That is, some of the nodes occur twice in a chromosome while some of them are missing from the same chromosome. As we observe in Fig. 5a, nodes 5 and 6 occur twice in Child 1 while node 3 is missing. On the other hand, in Child 2, nodes 5 and 6 are missing while node 3 occurs twice. We apply a repair procedure to make the invalid chromosomes valid. The procedure first removes the doubly listed nodes from the chromosomes then randomly assigns all the missing nodes to empty genes until all nodes are present in the chromosome once. After a crossover operation, n individuals in the population doubles to $2n$ individuals.

The next evolution process is mutation, as shown in Fig. 5b. In this process, we randomly swap the content of two genes, creating new individuals in the population. In the selection process, we select n individuals from the top of the population list

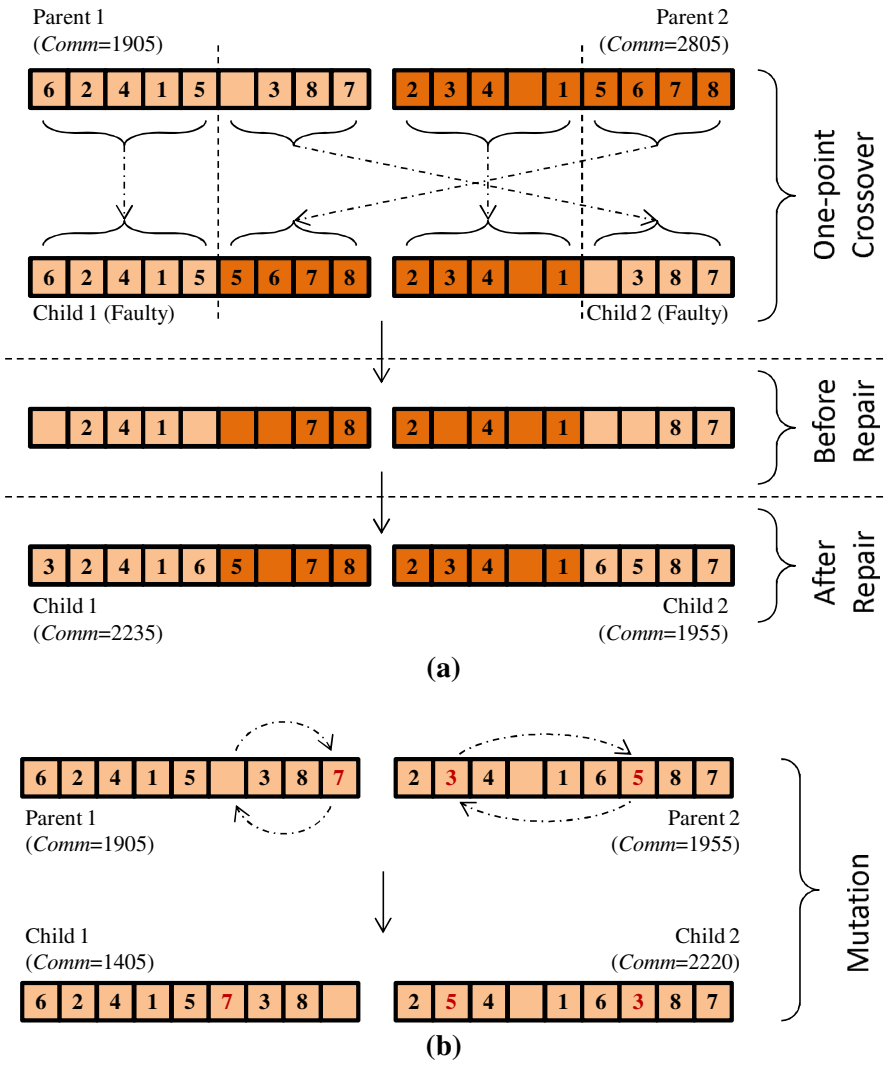


Fig. 5 **a** An example crossover operation. **b** An example mutation operation

based on their total communications and carry them to the next iteration. We apply the crossover and mutation operators until our stopping criterion is met. As our stopping criteria, we set the number of external loop iterations to $|P|^2$ as in SA-based method. We then select the solution with the best communication cost value. In Fig. 5, we give the fitness (i.e., Comm values) of each chromosome next to it.

Figure 6 plots the variation in the total communication in successive generations for WCTG given in Fig. 2a. As seen from this plot, our GA-based algorithm determines the optimum solution after a small number of population generations. Therefore, it is very eager to obtain the optimal solution.

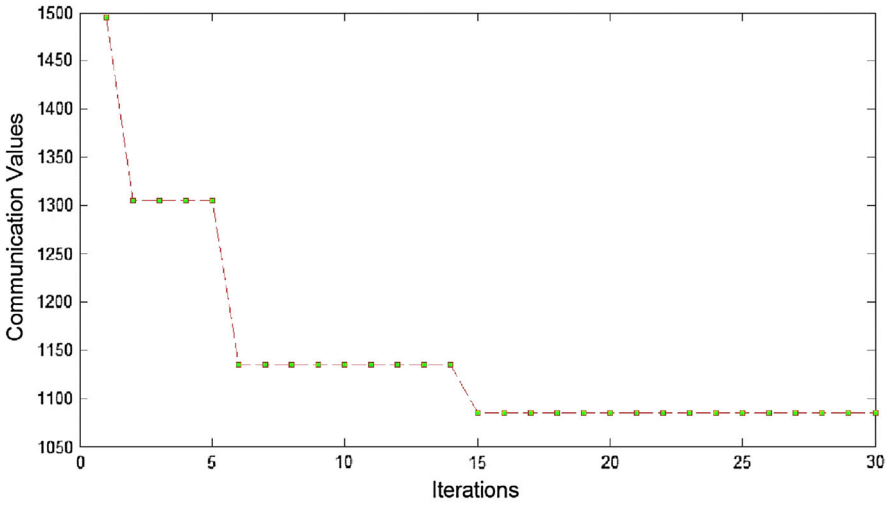


Fig. 6 An example iteration for accepted moves of GA-based algorithm

4.4 ILP-based method

Linear programming (LP) is an optimization method for problems that can be mathematically formulated by linear equalities and inequalities. An LP formulation consists of an objective function and one or more constraints that limit the solution space. ILP is a special case of LP, where the variables are restricted to having integer values. An ILP tool searches for the best result (i.e., the optimum solution) in this limited space using different methods, such as a simplex algorithm. The ILP optimization method is commonly used for networking, telecommunications, network flow, etc. If the ILP formulation fully covers the problem behavior, it determines the optimum solution. However, when the number of variables in the formulations increases, the solution times become unacceptable. Thus, ILP methods are good candidates for optimization problems with limited numbers of variables.

We formulated our mapping problem and solved it using Xpress-MP [28], a commercial tool. We presented our formulation in [23] and briefly explain it here for completeness. In our formulation, we used the 0–1 ILP. We present the variables and constant terms used in our formulation in Table 1.

In our ILP formulations, Eq. (11) ensures that each node is assigned to a tile, while Expression (12) states that some tiles can be empty (in a case where the number of nodes is lower than the number of tiles).

$$\sum_{x=0}^{X\text{dim}} \sum_{y=0}^{Y\text{dim}} \alpha_{i,x,y} = 1, \quad \forall i. \tag{11}$$

$$\sum_{i=1}^{n} \alpha_{i,x,y} \leq 1, \quad \forall x, y. \tag{12}$$

Table 1 Constants and variables used in the formulations

Constants and variables	Definitions
n	The number of nodes in WCTG
$w_{i,j}$	The communication weight between nodes i and j in WCTG
$X\text{dim}$	The size of the mesh architecture in the x dimension
$Y\text{dim}$	The size of the mesh architecture in the y dimension
$\alpha_{i,x,y}$	Binary variable. $\alpha_{i,x,y} = 1$ if task i is mapped to the tile in the coordinates (x, y) . Otherwise $\alpha_{i,x,y} = 0$
$X_{i,j,a}$	Binary variable. $X_{i,j,a} = 1$ if the distance in the x dimension between tasks i and j is equal to a. Otherwise, $X_{i,j,a} = 0$
$Y_{i,j,b}$	Binary variable. $Y_{i,j,b} = 1$ if the distance in the y dimension between tasks i and j is equal to b. Otherwise, $Y_{i,j,b} = 0$
$X\text{cost}$	The total communication cost of the network in the x dimension
$Y\text{cost}$	The total communication cost of the network in the y dimension

To determine the total communication cost, we must calculate the number of hops between two tasks mapped on the mesh, that is, we have to find the Manhattan distance [14] (i.e., city block distance) between two communicating tasks. We use Expression (13) and (14) to capture the distances a and b in dimensions x and y, respectively.

$$\begin{aligned} X\text{dist}_{i,j,a} &\geq \alpha_{i,x_i,y_i} + \alpha_{j,x_j,y_j} - 1, \quad \forall i, j.e_{i,j} \in E \\ 0 \leq x_i, x_j \leq X\text{dim}, \quad 0 \leq y_i, y_j \leq Y\text{dim} \;\; &\text{such that} \;\; a = |x_j - x_i| \end{aligned} \tag{13}$$

$$\begin{aligned} Y\text{dist}_{i,j,b} &\geq \alpha_{i,x_i,y_i} + \alpha_{j,x_j,y_j} - 1, \quad \forall i, j.e_{i,j} \in E \\ 0 \leq x_i, x_j \leq X\text{dim}, \quad 0 \leq y_i, y_j \leq Y\text{dim} \;\; &\text{such that} \;\; b = |y_j - y_i| \end{aligned} \tag{14}$$

Using (15) and (16), we then calculate the cost in the x and y dimensions. In these formulas, a and b represent the number of hops in the x and y dimensions, respectively. Multiplying these values with the communication weight, $w_{i,j}$, of two communicating tasks, i and j, gives us the total communication cost of these two nodes in the architecture:

$$X\text{cost} = \sum_{e_{i,j} \in E} \sum_{a=1}^{X\text{dim}} w_{i,j} \times a \times X\text{dist}_{i,j,a}. \tag{15}$$

$$Y\text{cost} = \sum_{e_{i,j} \in E} \sum_{b=1}^{Y\text{dim}} w_{i,j} \times b \times Y\text{dist}_{i,j,a}. \tag{16}$$

Consequently, our objective function can be expressed as:

$$\textbf{min} : \;\; \text{Comm} = X\text{cost} + Y\text{cost}. \tag{17}$$

We used the mapping returned from this formulation to calculate the overall energy consumption of the design.

4.5 CastNet: a Heuristic algorithm

Among the methods presented above, the ILP method guarantees the optimal solution; however, its runtime complexity is the highest. Thus, it does not produce the solutions in acceptable time limits for large problems. In our case, as we demonstrate in Sect. 5, when our WCTG exceeds 36 nodes, ILP cannot find an optimal solution in a tolerable time, which we set to 8-h time limit in our experiments. When it comes to accuracy, random mapping is the worst method among the candidates because there is no associated intelligence or improvement mechanism with it; each mapping is independent from the others. The SA and GA methods are metaheuristics and they rely on probabilistic decisions. They are common optimization techniques even though they do not guarantee optimum results.

For large optimization problems, CPU runtime is one of the major algorithm evaluation criteria. For example, in our mapping problem, determining the solution in a short time is very important since time-to-market is a crucial parameter in the integrated circuit field. In such a case, heuristic methods are good candidates for creating optimum solutions (or solutions that are good enough from the application's performance perspective). Heuristic methods use two different optimization techniques; constructive and iterative. In constructive heuristic algorithms, the solution is created step by step, applying decision criteria at each step. In iterative improvement cases, a solution is created and it is iteratively improved until a specified condition is met.

In this study, we use our constructive heuristic method called, CastNet [10]. Although CastNet is a constructive algorithm, it finds more than one solution, using the symmetry property of the mesh topology.

We give the pseudocode of the CastNet algorithm in Algorithm 2. In our algorithm, we initially select the first task to be mapped onto the mesh using the $highest\_priority(G)$ function (line 2). In this step, we assign priorities to the tasks based on their communication costs with their neighbors. We select the task with the highest priority as the first task to be mapped on the mesh. We next decide the initial cores, each of which is selected from different symmetry groups. Symmetry groups relate to the fact that some cores of the mesh are symmetrical. Thus, selecting one core from each symmetry group fulfills our purpose of covering all mapping results from the different initial cores. We use the $Candidates(M)$ function for this step (line 3). For each candidate (lines 5–18 in Algorithm 2), we first assign the initial task to the initial core. We then update the mapped (G') and unmapped (G) task sets. We accordingly update the mapped (T') and unmapped (T) tiles. We then select tasks based on the communication weights between mapped tasks using the function $node\_select(G, G')$. We also select the core for the selected task based on the communication cost on the mesh using the function $core\_select(T, T')$. We assign each selected task to the selected cores until no tasks remain in the unmapped task list. For each mapping, we calculate the communication cost and compare it with the previous mapping result. Finally, we find the best mapping with the minimum communication

Algorithm 2: Heuristic Mapping: CastNet

Data: G, M
Result: $T, E_{NoC}, Comm.$

begin
  $v_h = Highest\_Priority(G)$
  $C = Candidates(M)$
  $C_{best} = \infty$
  for $i \rightarrow 1$ **to** $|C|$ **do**
    $G_i = G; T_i = T; G_i^{'} = \emptyset; T_i^{'} = \emptyset$
    $f(v_h) = t_{C[i]}$
    $G_i = G_i - \{v_h\}; G_i^{'} = G_i^{'} + \{v_h\}$
    $T_i = T_i - \{t_{C[i]}\}; T_i^{'} = T_i^{'} + \{t_{C[i]}\}$
    while $G \neq \emptyset$ **do**
      $v_n = node\_select(G_i, G_i^{'})$
      $t_k = core\_select(T_i, T_i^{'})$
      $f(v_n) = t_k$
      $G_i = G_i - \{v_n\}; G_i^{'} = G_i^{'} + \{v_n\}$
      $T_i = T_i - \{t_k\}; T_i^{'} = T_i^{'} + \{t_k\}$
    $C_i = Calculate\_Comm(T_i)$
    if $C_i < C_{best}$ **then**
      $C_{best} = C_i; T_{best} = T_i$
  $T = T_{best}$
  $E_{NoC} = Calculate\_Energy(T_{best})$
  $Comm = C_{best}$
  return T, E_{NoC}, and $Comm$

cost and return it as our solution. We explain the implementation details of CastNet in [10].

5 Experimental results

5.1 Data set

We tested the aforementioned mapping methods on real multimedia benchmarks and on randomly generated task graphs. We selected six video applications from the literature, namely the video object plane decoder (VOPD) and the MPEG-4 decoder from [12], the multi-window display (MWD) from [29], and the 263 decoder (263 Dec.), 263 encoder (263 Enc.), and Mp3 encoder (Mp3 Enc.) from [30]. In the last experiment, we also used randomly generated graphs with varying number of nodes to test the scalability of the proposed methods.

5.2 Communication and energy minimization results

In Table 2, we present the results obtained by random, CastNet, SA, GA, and ILP mappings and give the total communication costs of the algorithms for all six

Table 2 Communication cost comparison of five mapping methods

Application	Number of vertices	Number of edges	Total communication (Mbit/s)				
			Random	GA	CastNet	SA	ILP
VOPD	16	20	6,668	4,290	4,135	4,290	4,119
MPEG-4	12	13	5,244	3,567	3,852	3,567	3,567
MWD	12	12	1,984	1184	1,248	1,184	1,184
263 Dec.	14	15	30.032	19.823	19.823	19.823	19.823
263 Enc.	12	12	317.051	230.407	230.407	230.407	230.407
Mp3 Enc.	13	13	28.4056	17.021	17.021	17.021	17.021

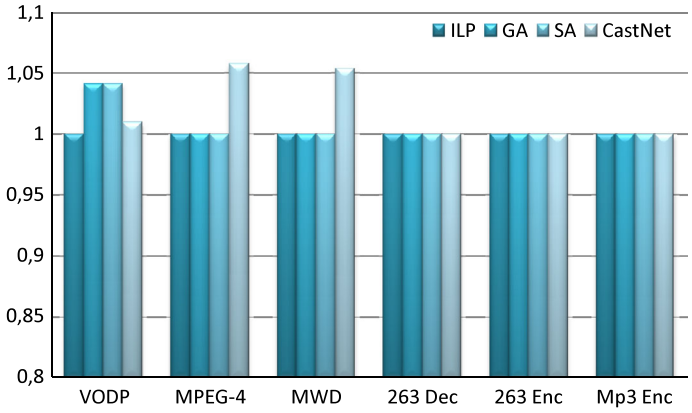
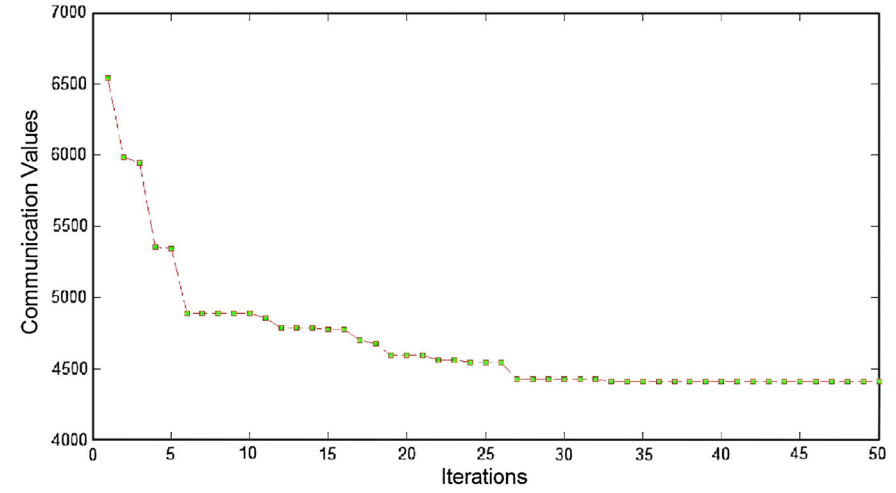


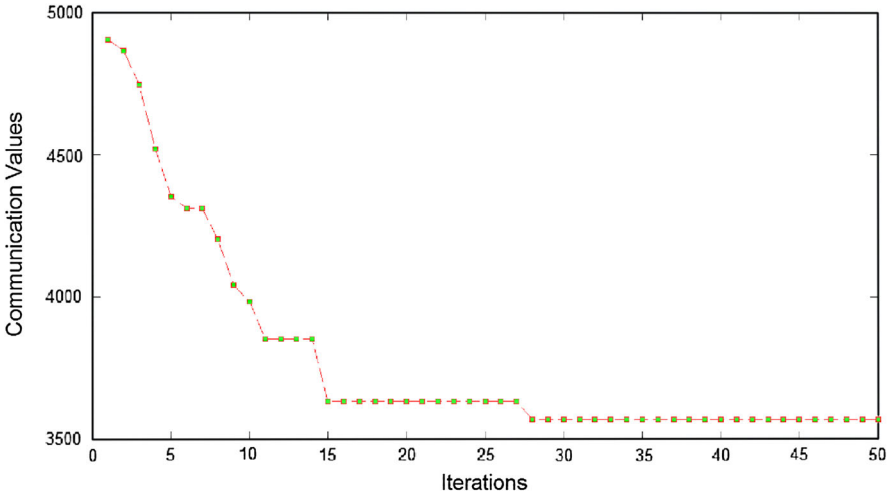
Fig. 7 Normalized energy consumption values with respect to results generated by ILP

multimedia benchmarks. While ILP obtains the best accuracy, it has the worst execution time values. We give the execution time analysis in the last experiment. The GA and SA methods obtain very similar results. As shown in Table 2, they determine optimum results for all benchmarks except VOPD; however, GA takes longer than SA to find the final result. CastNet obtains very promising results for these benchmarks when compared to SA and GA. In fact, for some of the benchmarks, it generates optimum results. It also obtained better result than GA and SA for VOPD benchmark.

We present our main objective criteria, energy consumption values, for the GA, SA, CastNet, and ILP methods in Fig. 7. In this bar chart, we normalize the energy consumption values of SA, GA, and CastNet against the results generated by ILP. For our energy model, we use the 100 nm technology power consumption parameters given in [30]. As can be observed from this comparison, GA and SA generate the same optimal results except VOPD. For VOPD benchmark, GA- and SA-based methods determines energy values close to 5 % to the optimal value while the energy values obtained by CastNet is within 1 % of the optimal result determined by ILP-based method. For MPEG-4 and MWD benchmarks, GA and SA determine the same energy values, which are better than CastNet's results. For these two benchmarks, the energy



(a) VOPD



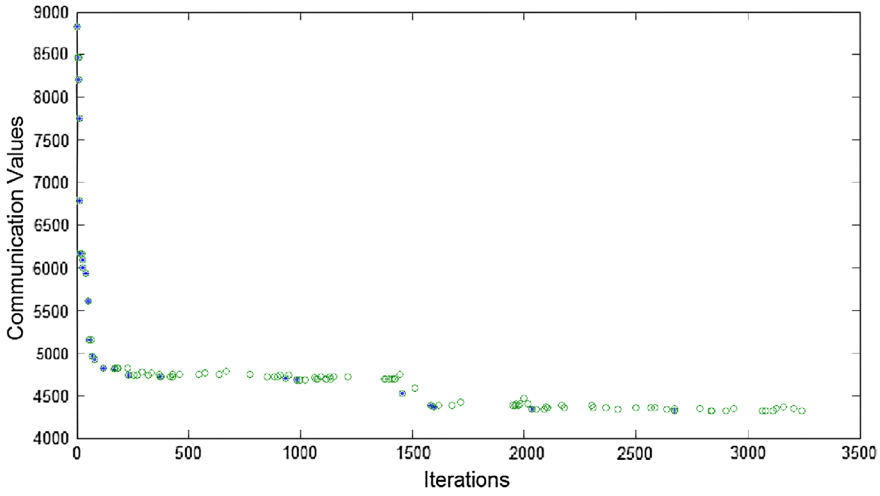
(b) MPEG-4

Fig. 8 Solution improvements of GA against iteration number for VOPD and MPEG-4 benchmarks

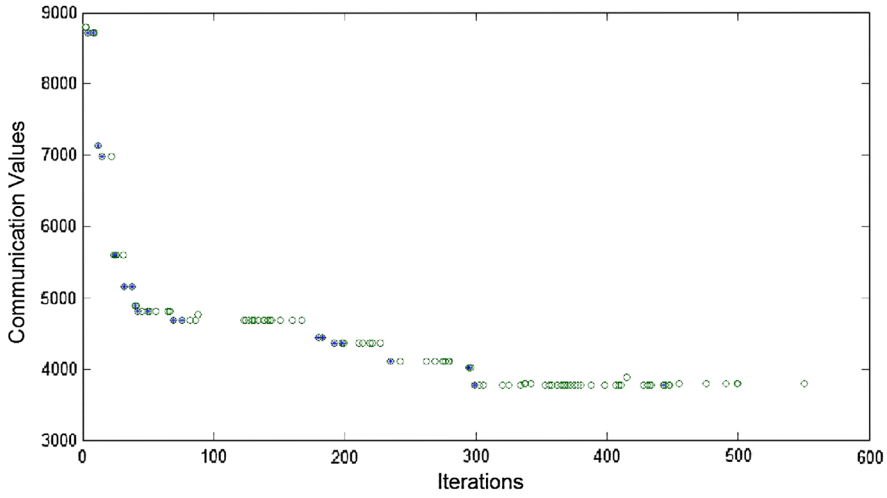
values determined by Castnet are 6 % far from the optimal result. For the last three benchmarks, all methods obtain the same optimal results.

5.3 GA and SA iterations

In Fig. 8, we give the best results of first 50 populations for genetic algorithm on VOPD and MPEG-4 benchmarks. The figures plot the total communication cost of each generation's best solution against the iteration of GA. As we can see, GA-based



(a) VOPD



(b) MPEG-4

Fig. 9 Solution improvements of SA algorithm against iteration number for VOPD and MPEG-4 benchmarks. Figures plot all accepted moves of SA procedure

mapping moves very fast towards the optimal solution by improving the current result in a few iterations.

In Fig. 9, we give the plots of all accepted moves for SA-based mapping algorithm on six benchmarks. In the figures, we show the first 1,000 iterations of SA and plot the total communication values of all accepted moves. As can be seen from these six plots, SA accepts some worse solutions than the current one at hand in an attempt to make up-hill moves to escape from local minima. When we compare the effectiveness of GA and SA to determine the optimal results, SA procedure consumes more iterations than GA to obtain the same results.

Fig. 10 WCTG of VOPD

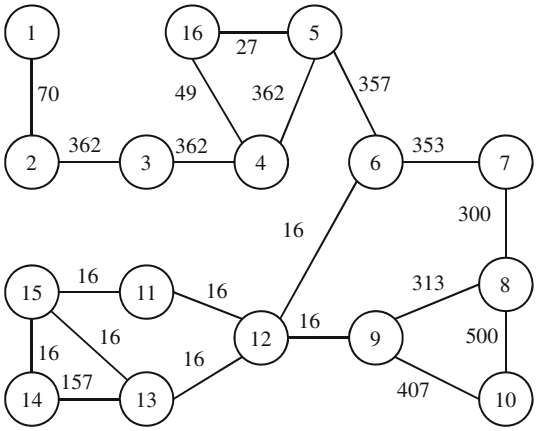


Table 3 Vertex descriptions of VOPD and MPEG-4 benchmarks

Vertex	VOPD	MPEG-4	Vertex	VOPD	MPEG-4
1	Var. length decoder	VU	9	Padding	ADSP
2	Run length decoder	AU	10	VOP memory	Up sample
3	Inverse scan Med.	CPU	11	Up sample	BAB
4	AC/DC prediction	RAST	12	Ref. memory	RISC
5	iQuant	sdRAM	13	Down sample	
6	IDCT	sRAM1	14	Arith. decoder	
7	Up sample	sRAM2	15	Memory	
8	VOP reconstruction	IDCT	16	Strip memory	

5.4 Example mappings

For illustrative purposes, we give the mappings of the VOPD and MPEG-4 benchmarks according to the WCTGs of VOPD and MPEG-4 given in Figs. 1 and 10, respectively. The vertex descriptions of these benchmarks are given in Table 3. We present the mappings generated by the GA-, SA-, CastNet-, and ILP-based methods for VOPD and MPEG-4 in Fig. 11. As can be observed from Fig. 11b, the communication costs of the mappings generated by GA, SA, and ILP are the same although the mappings look different. When we consider the symmetry of the mesh structure, however, the mappings are the same.

5.5 Test on random graphs

The number of vertices and edges are important parameters for the mapping problem because they heavily affect the mapping result and execution time. In our experiments presented above, we used multimedia benchmarks, whose numbers of vertices ranges between 12 and 16. However, to measure the effects of graph size, we should

GA
4290 Mbit/s

3	4	2	1
16	5	11	15
7	6	12	13
8	10	9	14

SA
4290 Mbit/s

8	7	16	3
10	6	5	4
9	12	11	2
14	13	15	1

CastNet
4135 Mbit/s

16	4	3	2
6	5	11	1
7	9	12	15
8	10	13	14

ILP
4119 Mbit/s

16	14	13	15
4	5	12	11
3	6	7	8
2	1	9	10

(a)

GA
3567 Mbit/s

2	4	3	6
1	5	10	9
	11	7	8
		12	

SA
3567 Mbit/s

	12	9	
8	7	11	2
	10	5	1
	6	4	3

CastNet
3852 Mbit/s

	9		
6	1	2	
4	5	8	3
11	10	7	12

ILP
3567 Mbit/s

		12	
2	10	7	8
4	5	11	9
6	1	3	

(b)

Fig. 11 Mappings of VOPD (**a**) and MPEG-4 (**b**)

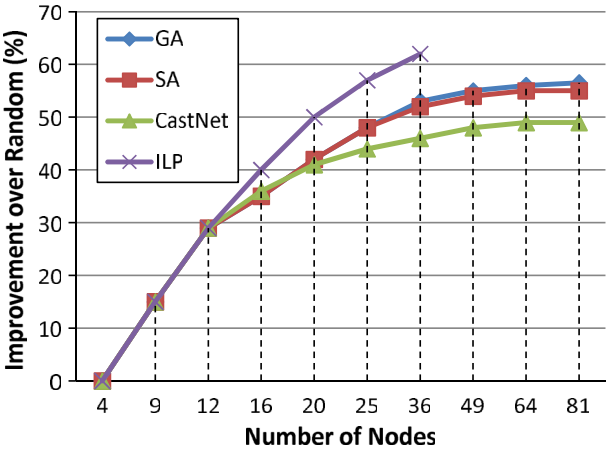


Fig. 12 Energy consumption improvements of ILP, CastNet, GA, and SA against random mappings

conduct experiments on graphs with different vertex degrees. Since the literature lacks benchmark graphs for NoC evaluation, we followed a commonly accepted method and randomly generated several edge-weighted graphs using the well-known random graph generator tool task graphs for free (TGFF) [31].

We mapped each generated graph on the smallest mesh topology that could contain all the nodes of the graph. In Fig. 12, we give the energy consumption comparisons of the ILP, CastNet, SA, and GA mapping algorithms against random mapping. In this comparison, the size of the graphs ranges from 4 to 81. For each graph, we generated 1,000 solutions for random mapping and selected the best one as our reference to

Table 4 Execution time comparison of five mapping methods

Application	Execution time (s)				
	Random	GA	CastNet	SA	ILP
VOPD	<0.1	55.41	<0.1	2.78	14,153
MPEG-4	<0.1	6.46	<0.1	0.25	7,750
MWD	<0.1	4.82	<0.1	0.16	380
263 Dec.	<0.1	21.74	<0.1	1.17	12,362
263 Enc.	<0.1	7.55	<0.1	2.13	5,825
Mp3 Enc.	<0.1	14.88	<0.1	0.18	8,476
Graph 20	<0.1	86	0.12	3.76	20,962
Graph 25	<0.1	154	0.25	5.45	26,068
Graph 36	<0.1	446	1.21	16.85	T.O.
Graph 49	<0.1	1,532	5.65	53.72	T.O.
Graph 64	<0.1	8,532	22	143	T.O.
Graph 81	<0.1	26,786	112	586	T.O.

compare with other mapping algorithms. Our ILP tool was not able to find solutions within 8 h when the number of nodes increases beyond 36. Below 36 nodes, ILP-based mapping obtained the optimal results, which can be used as reference points for the CastNet, GA, and SA algorithms. We can measure the differences between the optimum results from ILP and the results generated by these algorithms. As can be observed from the chart in Fig. 12, CastNet obtains better or similar results until the number of nodes is 20. After this point, GA and SA generate better solutions; however, this is only achieved through excessive execution latencies.

5.6 Performance evaluation

In the above experiments, we tested four optimization methods based on solution accuracy. Another important criterion to test these methods is the methods' execution times to determine the solution. We tested five mapping methods on six benchmark graphs and six randomly generated graphs with vertex sizes ranging from 20 to 81 nodes. In Table 4, we give the execution times of five mapping methods on the tested graphs. Among five mapping methods, ILP-based method obtains the optimum solutions when the number of nodes 25 or less. However, it cannot determine any solution to graphs with 36 nodes or more within the given time limit, which is set to 8 h. In Table 4, we place timeout (T.O.) if the method does not converge to a solution within 8 h.

Considering the random mapping is not an acceptable solution, our heuristic method CastNet determines the solutions in seconds for all graphs except the graph with 81 nodes. For graphs with large number of nodes, SA-based method is a better candidate than CastNet when our concern is accuracy. When the concern is execution time, SA-based method takes higher execution times than CastNet. When we compare three heuristics (GA, SA, and CastNet) based on execution times, the execution time of

GA-based method is the highest. For example, when the number of nodes is 81, GA barely finds the solution within eight hours. On the other hand, CastNet and SA take only minutes to determine the solutions.

6 Conclusion and discussion

In this paper, we presented several application mapping algorithms for mesh-based network-on-chip architectures to reduce energy consumption. These algorithms are based on ILP, GA, SA, and heuristic methods. We compared these algorithms against real multimedia benchmarks and against randomly generated task graphs.

Our experiments show that ILP obtains the optimum results; however, its runtime complexity is a drawback as it cannot generate a solution for graphs with more than 36 nodes. Similarly, graphs with more than 20 nodes took hours to solve.

The GA and SA optimization methods are based on probabilistic techniques. They start with randomly generated initial mappings and iteratively improve by swapping select nodes in the solution. GA works on a set of solutions to improve the best result, whereas SA tries to improve one solution iteratively. The complexity of GA is thus greater than that of SA; however, GA obtains better results most of the time (at the expense of higher execution latencies).

Heuristic methods are usually the fastest among the four methods presented in this paper. The presented mapping heuristic, called CastNet, obtains very promising results for the real applications. However, the number of nodes in these applications is fewer than 20. When the number of nodes in the application graphs increases, the results obtained by CastNet diverge from the optimum leaving the heuristic leadership to GA.

From these results, one can conclude that ILP is the best used for small- to medium-size applications, while GA or SA are best used for applications of any size, albeit with an increase in energy consumption. Heuristic methods can be used for mapping problems where execution time is crucial. Moreover, heuristic methods are especially important for NP-hard problems, such as the one presented in this paper.

Acknowledgments This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant Number 108E233 and 112E360.

References

1. Dally WJ, Towles B (2001) Route packets, not wires: on-chip interconnection networks. In: Proc. Design Automation Conference, Las Vegas, pp 684–689
2. Benini L, De Micheli G (2002) Networks on chips: a new SoC paradigm. IEEE Comput 35(1):70–78
3. Davis JA, Venkatesan R, Kaloyeros A, Beylansky M, Souri SJ, Banerjee K, Saraswat KC, Rahman A, Reif R, Meindl JD (2001) Interconnect limits on gigascale integration (GSI) in the 21st century. In: Proceeding of the IEEE, vol 89, no 3, pp 305–324
4. Sylvester D, Keutzer K (2000) A global wiring paradigm for deep submicron design. In: IEEE Transaction on computer-aided design of integrated circuits and systems (CAD/ICAS), vol 19, no 2, pp 242–252
5. Arabnia HR, Oliver MA (1989) A transputer network for fast operations on digitised images. In: International Journal of Eurographics Association (Computer Graphics Forum), vol 8, no 1, pp 3–12
6. Arabnia HR (1990) A parallel algorithm for the arbitrary rotation of digitized images using process-and-data-decomposition approach. J Parallel Distrib Comput 10(2):188–193

7. Bhandarkar SM, Arabnia HR (1995) The REFINE multiprocessor: theoretical properties and algorithms. Parallel Comput J 21(11):1783–1806
8. Arabnia HR, Smith JW (1993) A reconfigurable interconnection network for imaging operations and its implementation using a multi-stage switching box. In: Proceedings of the 7th Annual International High Performance Computing Conference. The 1993 High Performance Computing: New Horizons Supercomputing Symposium, Calgary, pp 349–357
9. Suleyman T, Ozcan O, Meltem O (2009) An ILP formulation for application mapping onto network-on-chips. In: 3rd International Conference on Application of Information and Communication Technologies, AICT2009, Baku
10. Suleyman T (2010) New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs. J Syst Architect. doi:10.1016/j.sysarc.2010.10.001
11. Murali S, De Micheli (2004) Bandwidth-constrained mapping of cores onto NoC Architectures. In: Proceedings of the Conference on Design, Automation and Test in Europe, vol 2, Washington DC
12. Janidarmian M, Khademzadeh A, Tavanpour M (2009) Onyx: a new heuristic bandwidth-constrained mapping of cores onto tile-based Network on Chip. In: IEICE Electron. Express, vol 6, no 1, pp 1–7
13. Shen W, Chao C, Lien Y, Wu A (2007) A new binomial mapping and optimization algorithm for reduced-complexity mesh-based on-chip network. In: Proceedings of the First international Symposium on Networks-on-Chip, Washington DC
14. Hu J, Marculescu R (2005) Communication and task scheduling of application-specific networks-on-chip. In: Computers and Digital Techniques, IEE Proceedings, vol 152, no 5, pp 643–651
15. Zhonghai L, Lei X, Axel J (2008) Cluster-based simulated annealing for mapping cores onto 2D mesh networks on chip. In: Proceedings of the 2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS '08), IEEE Computer Society, Washington DC
16. Moein-darbari Fahime, Khademzade Ahmad, Gharooni-fard Golnar (2009) CGMAP:a new approach to network-on-chip mapping problem. IEICE Electron Express 6(1):27–34
17. Ascia G, Catania V, Palesi (2004) Multi-objective mapping for mesh-based NoC architectures. In: Proceedings of the 2nd IEEE/ACM/IFIP international Conference on Hardware/Software Codesign and System Synthesis, Stockholm
18. Hung W, Addo-Quaye C, Theocharides T, Xie Y, Vijaykrishnan N, Irwin MJ (2004) Thermal-Aware IP Virtualization and Placement for Networks-on-Chip Architecture. Proceedings of the IEEE international Conference on Computer Design (October 11–13, 2004). ICCD. IEEE Computer Society, Washington, DC, pp 430–437
19. da Silva MVC, Nedjah N, Mourelle LM (2010) Power-aware multiobjective evolutionary optimisation for application mapping on network-on-chip platforms. Int J Electron 97(10):1163–1179
20. Nedjah N, da Silva MVC, Mourelle LM (2011) Customized computer-aided application mapping on NoC infrastructure using multi-objective optimization. J Syst Archit 57(1):79–94
21. Nedjah N, da Silva MVC, Mourelle LM (2012) Preference-based multi-objective evolutionary algorithms for power-aware application mapping on NoC platforms. Expert Syst Appl 39(3):2771–2782
22. Srinivasan K, Chatha, KS (2005) A technique for low energy mapping and routing in network-on-chip architectures. In: Proceedings of the 2005 international Symposium on low power electronics and design (San Diego, CA, USA, August 08–10, 2005). ISLPED '05. ACM, New York, pp 387–392
23. Tosun S (2011) Cluster-based application mapping method for network-on-chip. Adv Eng Softw 42(10):868–874
24. Marcon CAM, Moreno EI, Calazans NLV, Moraes FG (2008) Comparison of network-on-chip mapping algorithms targeting low energy consumption. Comput Digit Tech IET 2(6):471–482
25. Duato J, Yalamanchili S, Ni LM (2002) Interconnection networks: an engineering approach. Morgan Kaufmann
26. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
27. David EG (1989) Genetic algorithms in search, optimization, and machine learning, Addison-Wesley
28. http://www.dashoptimization.com
29. Chang K-C, Chen T-F (2008) Low-power algorithm for automatic topology generation for application-specific networks on chips. IET Comput Digit Tech 2(3):239–249
30. Srinivasan K, Chatha KS, Konjevod G (2006) Linear-programming-based techniques for synthesis of network-on-chip architectures. IEEE Trans Very Large Scale Integr Syst 14(4):407–420
31. http://ziyang.eecs.umich.edu/dickrp/tgff/