Journal of Systems Architecture 59 (2013) 361–371

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Application-Specific Network-on-Chip synthesis with flexible router Placement

J. Soumya *, Santanu Chattopadhyay

Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

ARTICLE INFO

Article history:
Received 25 August 2012
Received in revised form 2 March 2013
Accepted 20 May 2013
Available online 29 May 2013

Keywords:
Communication cost
Application Specific Network-on-Chip
Particle Swarm Optimization
Integer Linear Programming

ABSTRACT

Network-on-Chip (NoC) has been proposed as a possible solution to the communication problem in nano-scale System-on-Chip (SoC) design. NoC architectures with optimized application-specific topologies have been found to be superior to the regular architectures in designing Multi-Processor System-on-Chip (MPSoC) solutions. The application specific NoC design problem takes as input the chip floorplan, library of NoC components, and communication requirements between the tasks of the application. It outputs the positions of the routers in the floorplan, such that, all communication requirements of the application are satisfied. This paper presents an Integer Linear Programming formulation of the problem, followed by a heuristic technique based on Particle Swarm Optimization (PSO) for finding the router positions from the set of available positions within the chip floorplan. The goal is to minimize the communication cost between cores, satisfying both the link length and router port constraints. The results have been shown on realistic benchmarks. Comparisons have been carried out with regular mesh and custom architectures having routers positioned at (i) the corners of the cores, (ii) the centers of the cores, and (iii) the intersections of the cores. Significant reductions in communication cost have been observed over all the cases. For smaller benchmarks, the optimum results obtained via ILP matches exactly with those reported by the PSO. Many of the existing router placement policies fail even for these small benchmarks, when restrictions are imposed on permissible link length. This establishes the merit of the PSO formulation. Link and router energy consumption of the synthesized NoC have been compared with regular mesh based architectures. The results show significant reduction in communication cost, area overhead, link energy and router energy in the synthesized NoC over regular mesh topology as well.

1. Introduction

With advancement in the nanometer era, number of cores in a single chip and associated communication complexity are increasing rapidly. To achieve high communication efficiency, Network-on-Chip (NoC) has been proposed [1]. In NoC, communication among various cores is achieved through on-chip micro-network components (such as switches and network interfaces), instead of the traditional non-scalable buses. NoCs can be designed using either regular or application-specific network topologies. For regular topologies, some existing NoC solutions assume a mesh architecture [2–4]. Although standard topologies can easily be implemented and also ensure the regularity of the on-chip network, they might not be the best choice for Application-Specific Network-on-Chip (ASNoC) due to mismatch between the topology and the application characteristics [5]. Custom topologies are needed to handle the challenges in terms of irregular core sizes, different core locations, and communication flow requirements. However, the approaches followed in the literature for the custom topology based NoC synthesis do not consider the router placement problem in the synthesis process. It has been observed in [6] that core and/or router placement problem in NoC is NP-hard. As a result, most of the works have conveniently assumed the router associated with a core to be placed at a fixed position within the rectangular tile containing the core. Some such fixed locations are, corners [26,27], centers [29] and intersections [34] within the floorplan. This may necessitate long interconnects or multiple hops for a communication between cores. This has motivated us to solve the problem of determining the best possible router locations in the floorplan as part of the synthesis problem.

Consider the following given as input to the application specific NoC synthesis problem.

(1) The directed graph $G(C, E)$ corresponding to the communication traces of the application.

   (a) Each vertex $c_i \in C$ corresponds to a core in the core-graph.

* Corresponding author. Tel.: +91 9434139447.
E-mail addresses: soumyaj@ece.iitkgp.ernet.in (J. Soumya), santanu@ece.iitkgp.ernet.in (S. Chattopadhyay).

1383-7621/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2013.05.013

(b) An edge e_{ij} represents communication from core c_i to core c_j. The edge is labeled with a value BW_{ij} equal to the bandwidth requirement of the communication from c_i to c_j.

(2) A floorplan showing the positions and dimensions of cores within the NoC.

(3) The set of probable router positions.

(4) A constraint L_{MAX} on the maximum permitted interconnect length between a core and the associated router, and also between the routers.

The NoC synthesis procedure assigns exactly one router to each core. The routers are placed at a subset of possible router positions, given as input. Links needed to accomplish communication between cores are identified, satisfying link-length constraint L_{MAX}.

The overall communication cost [3] of the NoC is defined to be the sum of communication costs for each pair of cores in it. For a pair of cores c_i and c_j, the communication cost is the product of the bandwidth requirement of their communication and the hop distance between their associated routers. The routers are to be placed, such that, this overall communication cost is minimized. We also consider that the individual routers are much smaller in size than the cores. This assumption is supported by [1] in which it has been observed that, on an average, the entire NoC places an area overhead of 6.6% on the SoC architecture.

In this paper, we focus on the application-specific NoC synthesis by determining the best possible positions for the routers (unlike fixing at some location in the floorplan [20,22,26–29,34]) from the set of available positions. We have first developed an Integer Linear Programming (ILP) formulation of the router placement problem. It gives exact solution, but works for only small-sized problems. For larger application graphs, a Particle Swarm Optimization (PSO) based solution has been proposed. The rest of the paper is organized as follows. Section 2 presents a motivating example for our case. Section 3 reviews the existing literature. Section 4 gives the NoC synthesis flow. Section 5 details the ILP formulation. Section 6 gives an overview of PSO. Section 7 contains the formulation of PSO for selecting the router positions. Section 8 enumerates the experimental results. Section 9 concludes the paper.

2. A motivating example

The advantage of having the flexibility in choosing router locations compared to placing them at corners can be understood by the example noted in Fig. 1. It corresponds to the benchmark application PIP, having eight cores *C0* through *C7*.

Fig. 1a shows the communication trace graph of the application, in which the edges are annotated with bandwidth requirements between corresponding tasks in Mb/s. Fig. 1b shows a floorplan for the cores. It also includes the routers (shown as circles) at corners corresponding to the cores. Fig. 1c shows the floorplan with router locations found by the approach proposed in this paper (It may be noted that the floorplan is an input to the router location problem solved in this paper, we do not generate the floorplan). When the routers are located at fixed corners of tiles containing cores, it may require long interconnects between the routers, or multiple hops for a communication between cores. For example, let us consider the communication between *core3* and *core6*. For Fig. 1b, that places routers only at corners, it requires either a long interconnect or needs to travel in three hops (using routers attached with cores *core3*, *core5*, *core4*, and *core6*). On the other hand, in our approach (Fig. 1c), the routers are placed on the floorplan such that smaller interconnects can be used between the routers. Core-to-core communications also need less number of hops in this case. For example, the same communication from *core3* to *core6* requires only a single hop now. This happens due to the intelligent placement of routers that also considers the link length constraint put into the synthesis process. In this case, we could obtain a 35.71% decrease in the overall communication cost.

3. Related work

A survey of ASNoC design techniques has been presented in [4]. It enumerates the advantages of custom topologies over standard ones for ASNoC. While some of the design techniques attempt to modify a regular topology to take into consideration the variation in core sizes and communication pattern, others try to synthesize the most suited topology. A performance aware design methodology of inserting long-range interconnects on top of a regular mesh based network for synthesis of application specific architectures has been presented in [7]. In the following we review the major ASNoC synthesis works.

First, we will look into the strategies that do not consider the floorplan of the NoC. A heuristic has been proposed in [8] for constraint driven communication synthesis of on-chip networks. They have proposed a quadratic programming approach and a clustering algorithm for this purpose. However, no floorplan information is utilized during the topology generation process. A method to reduce hardware cost of NoC via link aggregation has been presented in [9]. It creates application specific NoC with non-uniform distribution of physical links in logical connections, depending on the amount of network traffic that is passed through them. A network partitioning technique based on Fiduccia–Mattheyses (FM) partitioning algorithm has been proposed in [10] to reduce the area cost. A linear programming based technique has been proposed in [11] to generate an area optimized NoC architecture constrained to propagation delay time and bandwidth requirements. Different topology generation techniques have been analyzed and compared in [12] with respect to area and average delay. They have analyzed network partitioning techniques, long-range generation technique and thus determine the best scheme to be used for NoC topology generation. A tool, *Q8WARE* has been presented in [13], which is a small scale version of ASIC assembly line. It deals with synthesis of application specific NoC with hardware reusability constraints. A genetic algorithm based technique has been presented in [14] for synthesis of custom NoC architectures that support guaranteed throughput traffic. The energy consumption of NoC is optimized by minimizing the cumulative traffic flowing through the ports of all routers. The total area consumption is minimized by reducing the total number of routers used. A methodology for generating energy efficient application specific architectures has been presented in [15]. It uses application partitioning to reduce processing element dependencies. Topology exploration and buffer sizing techniques are utilized to generate custom topology with reduced power consumption. A NoC topology generation and analysis method has been presented in [16] that addresses throughput requirements by considering the latency present in the system. An irregular application specific topology generation algorithm has been presented in [17] to reduce power consumption. It clusters the given application based on the communication characteristics and then construct the topology by connecting clusters to each other one by one.

Next we look into the floorplan aware methodologies. A custom NoC instantiation tool *Xpipes Compiler* has been presented in [18], which is based on designer specified inputs. A floorplan-aware tool has been presented in [19] for NoC design and synthesis integrated with a graphical user interface for interacting among abstract traffic flow specification, topology synthesis and floorplanning. A Genetic Algorithm (GA) based technique has been presented in

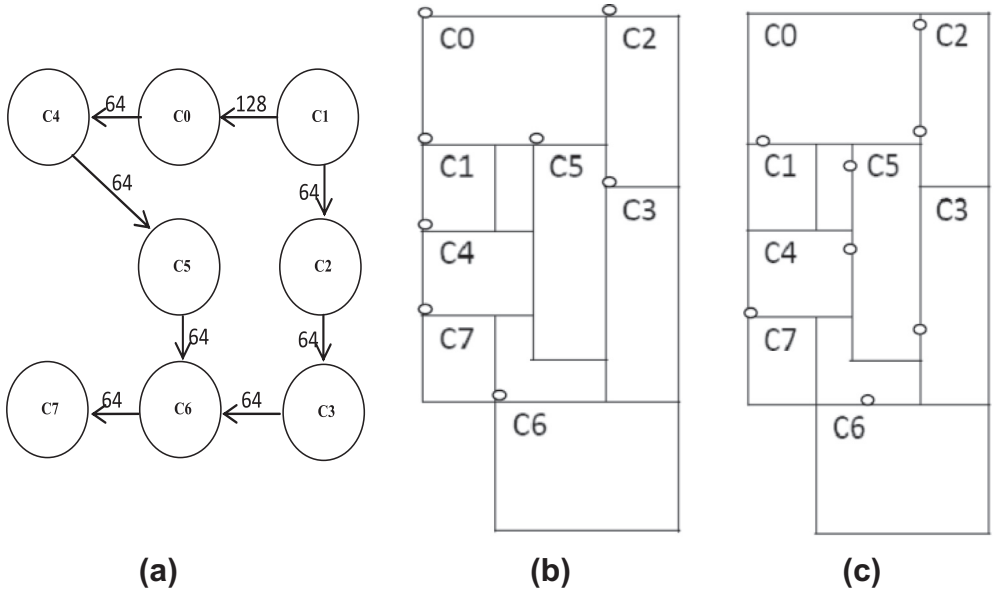

Fig. 1. PIP: (a) Communication trace graph, (b) floorplan with routers at corners and (c) floorplan with router positions using our approach.

[20] for ASNoC design with an objective of minimizing the power consumption. The work assumes the routers to be at corners of the cores. A branch and bound algorithm has been proposed in [21] for customized communication architecture synthesis. It uses the core coordinates from the floorplan. GA based topology synthesis method has been proposed in [22] to minimize power consumption by taking floorplan information. This work also assumes the routers to be at the corners of the cores. In [23], multi-objective approach to topology design based on Tabu search technique has been presented to meet power and performance requirements of an application. A synthesis oriented design flow, *Xpipes lite* has been presented in [24] for the generation of synthesizable and simultaneous models for application specific NoCs.

In [25], a physical floorplan has been used during topology design to reduce power consumption on wires. However, the area and power consumption of the switches are not taken into consideration. In [26], a slicing tree based floorplanner has been used during the topology design process. This work assumes that the switches to be located at a corner of the cores, the network components are not considered in the floorplanning process. In [27], NoC topology generation algorithms have been presented based on slicing structures where switch locations are restricted to corners of cores. In [28], a two step topology generation procedure has been proposed using a min-cut partitioner to cluster highly communicating cores on the same switch and a path allocation algorithm to connect clusters together to minimize power consumption. In [29], an iterative refinement strategy to generate an optimized NoC topology has been presented that supports both packet switched network and point to point communications. This assumes the network interfaces for the processing cores to be located on the corner, while router nodes are in center. In [30], a partition driven floorplanning algorithm has been proposed that uses a heuristic to insert switches and an algorithm for inserting NIs, limited to mosaic type of floorplans. In [31], two heuristic algorithms have been proposed to examine different set partitions. Partitioning is carried out based on communication flow and a physical network topology has to be generated for each partition. A three stage synthesis approach has been presented in [32] that integrates communication requirements, physical information among cores, and partitioning, into the floorplanning phase to explore the optimal switch number for clustering of cores with minimized link and switch power consumption. A complete synthesis flow has been illustrated in [33] for customized NoC architectures which partitions the flow into major steps of topology mapping, selection and generation. It provides tools namely, *SUNMAP*, *Xpipes Compiler* for their automatic execution. Thermal and non-thermal aware application specific NoC synthesis framework has been presented in [34] that combine multiple algorithms and heuristics to efficiently explore the solution space. The works have described both thermal and non thermal aware approaches for router placement. Both techniques assume that switches can only be placed at the interconnection of cores in the floorplan. A topology generation method has been presented in [35] by employing analytical models and simulation tools to design low power, high performance custom NoCs. The above works generate floorplan as part of the synthesis process.

All the approaches reported in the literature either do not consider the floorplan information in the synthesis process or assume that the router locations are fixed at some point in the floorplan. Thus, to the best of our knowledge, ours is the first work which addresses the problem of flexible router placement for minimizing the communication cost of the application. Next section will give an overview of our approach.

4. NoC synthesis flow

As mentioned in Section 1, we assume that the cores corresponding to the tasks of the application are already laid out on a two-dimensional grid, corresponding to the floorplan of the chip. Now, the routers will be inserted into this grid floorplan of the application. Individual routers are assumed to be of size equal to the smallest square in the grid. This is a valid assumption as the routers are in general, much smaller than the cores.

All routers are taken to be of same size. This constraint can of course be relaxed if we assume that the routers are of different complexities and thus have different area requirements. The maximum allowed link-length (L_{MAX}) is taken as another constraint. Thus, if a router is located at grid point (i, j) and a second one at (k, m), a link can exist between the two, if the distance $\sqrt{((i-k)^2+(j-m)^2)}$ is less than L_{MAX}. The flow of the approach is presented in Fig. 2. It may be noted that depending upon the pin

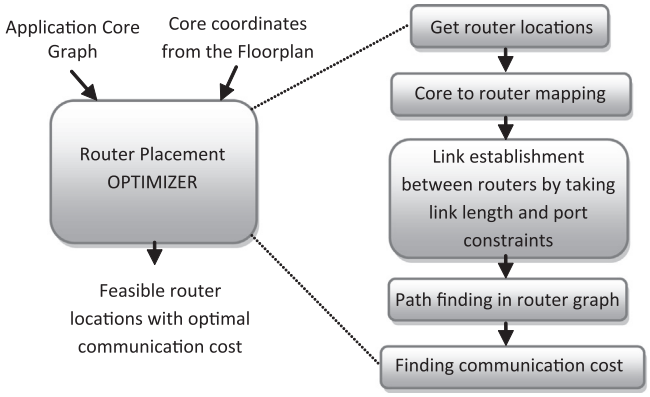

Fig. 2. NoC synthesis flow.

distribution of individual cores, more constraints can be placed about the probable router locations in the floorplan. Our approach takes as input these possible router locations and determines the final router locations to optimize communication cost.

5. ILP based approach

This section presents an ILP formulation of the problem of finding suitable router locations in a given NoC floorplan to optimize communication cost.

5.1. Parameters and Variables

The parameters and variables used in the ILP formulation are as follows.

Parameters and Variables	Definitions
$rdist_{r_i r_j}$	The pre-calculated distance between probable router locations r_i and r_j
$cdist_{c_i r_s}$	The pre-calculated distance between core c_i and router location r_s
L_{MAX}	The maximum link length that is allowed between two routers as well as between a core and a router
BW_{ij}	The bandwidth requirement from core c_i to core c_j
sr_s	Binary variable. $sr_s = 1$ if location r_s is selected to contain a router. Otherwise, $sr_s = 0$
$m_{c_i}^{r_s}$	Binary variable. $m_{c_i}^{r_s} = 1$ if core c_i is mapped to the router at r_s. Otherwise, $m_{c_i}^{r_s} = 0$
$cl_{c_i}^{r_s}$	Binary variable. $cl_{c_i}^{r_s} = 1$ if link exists between core c_i and router location r_s. Otherwise, $cl_{c_i}^{r_s} = 0$
$rl_{r_i r_j}$	Binary variable. $rl_{r_i r_j} = 1$ if link exists between router locations r_i and r_j, that is, distance between r_i and r_j is less than L_{MAX}. Otherwise, $rl_{r_i r_j} = 0$
$P_{c_i c_j}^{r_s r_t}$	Binary variable. $P_{c_i c_j}^{r_s r_t} = 1$ if a path exists between routers r_s and r_t to which cores c_i and c_j have been mapped, respectively. Otherwise, $P_{c_i c_j}^{r_s r_t} = 0$
$n_i^{r_s r_t}$	Binary variable. $n_i^{r_s r_t} = 1$ if router i is part of the path from router r_s to r_t. $i \in N(r_s, r_t)$, where $N(r_s, r_t)$ is the set of routers along the path from r_s to r_t. Otherwise, $n_i^{r_s r_t} = 0$
$l_{r_i r_j}^{r_s r_t}$	Binary variable. $l_{r_i r_j}^{r_s r_t} = 1$ if link between routers r_i and r_j is part of the path from r_s to r_t. Otherwise, $l_{r_i r_j}^{r_s r_t} = 0$
$D_{r_s r_t}$	Integer variable. The distance between two router locations r_s and r_t in terms of number of hops. The value of this variable can be in the range of 0 to total number of routers in the network

5.2. Objective Function

Our objective is to minimize the communication cost by selecting suitable router locations. The objective function can be formulated as follows. If cores c_i and c_j are mapped to router locations r_s and r_t and a path exists between them in the network, $P_{c_i c_j}^{r_s r_t}$ is equal to 1. This multiplied by $D_{r_s r_t}$ gives the number of hops of the communication from c_i to c_j. Number of hops multiplied by bandwidth gives the communication cost, which has to be minimized over all the edges in the core graph. Let R be the set of routers, C be the set of cores, and E be the set of edges between cores. Hence, the overall objective function is,

$$\text{Minimize}\left[\sum_{e_{ij}\in E} BW_{ij}\left(\sum_{r_s, r_t \in R} D_{r_s r_t} \times P_{c_i c_j}^{r_s r_t}\right)\right] \quad (1)$$

5.3. Constraints

The following is the set of constraints framed to solve the location selection problem.

(1) Mapping constrains

- Each core has to be mapped onto only one router.

$$\forall c_i \in C, \sum_{r_s \in R} m_{c_i}^{r_s} = 1 \quad (2)$$

- Similarly, each router is selected for synthesis should have one core mapped onto it.

$$\forall r_s \in R, \sum_{c_i \in C} m_{c_i}^{r_s} = 1 \quad (3)$$

- Finally, a core can be mapped to a router only when that router location is selected for synthesis.

$$\forall r_s \in R, sr_s - m_{c_i}^{r_s} \geqslant 0 \quad (4)$$

(2) Router–core association constraints

- A core c_i can be associated to the router at location r_s (that is $cl_{c_i}^{r_s}$ can be set to 1) if the physical distance $cdist_{c_i r_s}$ is less than L_{MAX}.

$$\forall c_i \in C, \forall r_s \in R, cdist_{c_i r_s} \times cl_{c_i}^{r_s} \leqslant L_{MAX} \quad (5)$$

- Now, mapping of a core onto a router is possible if there is a link between them.

$$\forall c_i \in C, \forall r_s \in R, cl_{c_i}^{r_s} - m_{c_i}^{r_s} \geqslant 0 \quad (6)$$

(3) Inter-router link constraints

- Link can exist between routers, if the distance between them is less than the maximum link length L_{MAX}.

$$\forall r_i \in R, \forall r_j \in R, i \neq j, rdist_{r_i r_j} \times rl_{r_i r_j} \leqslant L_{MAX} \quad (7)$$

(4) Constraints for core graph edges
- Each edge present in the core graph has to be mapped onto a path in the evolved topology.

$$\forall e_{ij} \in E, \forall r_s, r_t \in R, m_{c_i}^{r_s} + m_{c_j}^{r_t} - P_{c_i c_j}^{r_s r_t} \leqslant 1 \quad (8)$$

$$and,\ 2 \times P_{c_i c_j}^{r_s r_t} \leqslant m_{c_i}^{r_s} + m_{c_j}^{r_t} \quad (9)$$

(5) Path constraints

To find the path between two router locations r_s and r_t, we have to know which of the links and routers are part of the path. We impose the following conditions for this purpose.
- Starting and ending routers must be part of the path.

$$\forall e_{ij} \in E, \forall r_s, r_t \in R, P_{c_i c_j}^{r_s r_t} - n_i^{r_s r_t} = 0 \quad (10)$$

- To get starting link of the path, we impose the following constraint. Starting router may have more than one outgoing links, out of which exactly one has to be selected.

$$\forall e_{ij} \in E, \forall r_s, r_t \in R, P_{c_i c_j}^{r_s r_t} - \sum_{r_i, r_j \in R} l_{r_i r_j}^{r_s r_t} = 0 \quad (11)$$

- If a link is part of the path, starting and ending routers of that link must also be part of the path.

$$\forall r_s, r_t \in R, \forall r_i, r_j \in R, 2 \times l_{r_i r_j}^{r_s r_t} - n_i^{r_s r_t} - n_j^{r_s r_t} \leqslant 0 \quad (12)$$

- Each router (except starting and ending one) should have equal in and out degrees.

$$\forall r_s, r_t \in R, \forall r_i \in R, r_i \neq r_s, r_i \neq r_t, r_i \neq r_j, 2 \times n_i^{r_s r_t} - \sum_{\forall (r_i, r_j) \in R} l_{r_i r_j}^{r_s r_t} = 0 \quad (13)$$

- A link can be part of the path if it satisfies link length constraint.

$$\forall r_s, r_t \in R, \forall r_i, r_j \in R, rl_{r_i r_j} - l_{r_i r_j}^{r_s r_t} \geqslant 0 \quad (14)$$

- The distance between routers can be calculated by using following equation.

$$\forall r_s, r_t \in R, D_{r_s r_t} - \sum_{\forall (r_i, r_j) \in R} l_{r_i r_j}^{r_s r_t} = 0 \quad (15)$$

This completes the formulation. The objective function along with the constraint set can be fed to any ILP solver to get the router positions minimizing the communication cost for the synthesized NoC. We have used the tool CPLEX [36] for this purpose. However, excepting for very small NoCs, it takes huge amount of CPU time to arrive at the solution. Hence we have resorted to a PSO based optimizer to explore the search space for synthesizing larger NoCs.

6. Particle Swarm Optimization

Particle Swarm Optimization (PSO) [37] is a population based stochastic technique developed by *Eberhart* and *Kennedy* in 1995, inspired by social behavior of bird flocking or fish schooling. In a *PSO* system, multiple candidate solutions coexist and collaborate simultaneously. Each solution, called a *particle*, flies in the problem space according to its own experience as well as the experience of neighboring particles. It has been successfully applied in many problem areas. In *PSO*, each single solution is a particle in the search space, having a fitness value. The quality of a particle is evaluated by its fitness. Inspired by its success in solving problems in continuous domain, several researchers have attempted to apply it in discrete domain as well [38]. This has motivated us to look for a DPSO formulation of finding suitable router positions as part of the NoC synthesis problem. In this section we present a brief overview of the *DPSO* strategy.

Let the position of a particle (in an n-dimensional space) at k^{th} iteration be $p_k =< p_{k,1}, p_{k.2}, \ldots p_{k,n} >$. For i^{th} particle, the quantity is denoted as p_k^i. Let $pbest^i$ be the local best solution that particle i has seen so far over the generations, and $gbest_k$ be the global best particle (solution) of generation k. The new position of particle i is calculated as follows:

$$p_{k+1}^i = (c_1 * I \oplus c_2 * (p_k \rightarrow pbest^i) \oplus c_3 * (p_k \rightarrow gbest_k)).p_k^i \quad (16)$$

In the above expressions, $a \rightarrow b$ represents the minimum length sequence of swapping to be applied on components of a to transform it to b. For example, if a = <1,3,4,2> and b = <2,1,3,4>, $a \rightarrow b$ = <*swap* (1,4), *swap* (2,4), *swap* (3,4)>. The operator $\oplus$ is the fusion operator. Applied on two swap sequences, $a \oplus b$ is equal to the sequence in which the sequence of swaps in a is followed by the sequence of swaps in b. The constants c_1, c_2, c_3 are the inertia, self-confidence and swarm confidence values. The quantity $c_i * (a \rightarrow b)$ means that the swaps in the sequence $a \rightarrow b$ will be applied with a probability c_i. I is the sequence of identity swaps, such as, <*swap* (1,1), *swap* (2,2), $\cdots$ *swap* (n, n)>. It corresponds to the inertia of the particle to maintain its current configuration. The final swap corresponding to $c_1 * I \oplus c_2 * (p_k \rightarrow pbest^i) \oplus c_3 * (p_k \rightarrow gbest_k)$ is applied on particle p_k^i to generate p_{k+1}^i.

From [39], it can be found that the convergence condition for this DPSO is given by,

$$(1 - \sqrt{c_1})^2 \leqslant c_2 + c_3 \leqslant (1 + \sqrt{c_1})^2 \quad (17)$$

Accordingly, we have worked with various values of $c_1, c_2 and c_3$. The results reported in this paper are based upon the values $c_1 = 1, c_2 = 0.5, c_3 = 0.5$.

7. Minimization of communication cost via PSO

In this section we present our formulation to select the router positions, such that the communication cost is minimized. Input to our formulation is a floorplan of the chip for the application under consideration and all available router positions.

7.1. Particle Formulation and Fitness Function

We first enumerate the structure of a particle. We assume the number of routers to be equal to the number of cores present in the application. Let l be the number of available router positions in the floorplan. For all these l available router positions, a particle is a permutation of numbers from 0 to $l - 1$. It is assumed that the router positions are numbered as 0 to $l - 1$. A particle identifies a set of router positions. The total communication cost forms the fitness function. While calculating the communication cost, we consider the locations from 0 to n ($n < l$) in the particle, where, n represents the number of cores in the application. Thus, we use only those first n router positions for the mapping of cores.

Fitness of a particle P
= Total communication cost due to the router positions specified by the particle.

For every particle, the following steps are followed to calculate its fitness value:

(1) Distance between each pair of core and router is calculated.
(2) A core is mapped to its nearest router. If the distance is more than L_{MAX}, the fitness of the particle is set to infinity.
(3) Links are established between the routers, taking L_{MAX} constraint into account. No link can be of length larger than this.

(4) For each edge in the core graph, shortest path is found between the cores in the router graph and the corresponding communication cost is computed.

It may be noted that while identifying the shortest paths between the cores, we have taken into consideration the capacities of constituent links and all the communications passing through the link. The issues like deadlock can be taken care of later either by adding virtual channels or via communication scheduling.

7.2. Local and Global bests

Every particle has a *local best* (*pbest*), which is one set of router positions giving minimum communication cost, among all sets of router positions that the particle has seen so far in the evolution process. This local best partially guides the evolution of the particle. For a particular generation, the *global best* (*gbest*) is the particle resulting in the minimum communication cost for that generation. It also controls the evolution of particles. The local best of each particle and the global best are modified if the corresponding values in the current iteration are less than the values till the previous iteration.

7.3. Evolution of generation

Evolution of the particles is done over generations to create new particles which are expected to give results closer to the optimum. To start with, the initial population is created randomly and the fitness of each particle is evaluated. The local best (*pbest*) of each particle is initialized to be the same as that of the initial particle. The global best of the generation is set to the particle giving the least communication cost (smallest fitness function) in the generation. The second generation results through random exchange of router positions within the particles. The local best and the global best values are updated if they give better fitness values. The further generations are created through a series of *swap* operations. The local best of each particle and the global best are modified if the corresponding values in the current generation are less than the values in the previous generation. The local best and the global best evolution thus centers on the basic operator, *swap*, explained next.

7.3.1. Swap operator

Each particle is a sequence of l probable router positions. Out of these, first n correspond to the selected router positions. To effect a change in the particle, the swap operator is used. The operator takes two indices (say, i and j) of particle P as input and creates a new particle P_1. The particle P_1 is same as P excepting that the positions i and j of P are interchanged in it.

Let the particle P be

r1	r3	r5	**r7**	r4	**r2**	r6	r8

where *rx* represents the router at position *x*. For example, the indices of *r1*, *r3*, *r5* are 0, 1 and 2 respectively. The swap operator SO(3,5) swaps positions 3 and 5 in P to generate a new particle as shown below.

r1	r3	r5	**r2**	r4	**r7**	r6	r8

7.3.2. Swap sequence

A swap sequence is a sequence of swap operators. For example, a swap sequence SS = {SO(1,7), SO(3,4)} creates particle P_{new} working on particle P in two steps as follows.

Particle P:

r3	**r6**	r8	r4	r1	r5	r2	**r7**

SO (1,7) on particle P creates an intermediate particle P_{int}

P_{int}:

r3	**r7**	r8	r4	r1	r5	r2	**r6**

SO(3,4) on P_{int} results in the new particle P_{new}

P_{new}:

r3	r7	r8	**r1**	**r4**	r5	r2	r6

After all particles have undergone the evolution, a new generation gets created. The best fitness of this generation gives the global best for the population. The PSO terminates if there is no improvement in *gbest* value for a pre-defined number of generations, or, the PSO has already iterated for a preset maximum number of generations.

8. Experimental results

In this section we present our results on effective choice of router locations in NoC. For this, we have worked with benchmark NoCs noted in Table 1. Since the core dimensions for them are not available in the literature, we have assumed the values noted in the Table. It may be mentioned that the work [40] suggests the core dimensions to vary between $1 \times 1\ \text{mm}^2$ to $3 \times 3\ \text{mm}^2$. Floorplans are generated randomly for each application. The value of L_{MAX} is taken to be 2.5 mm, as increasing the link length needs insertion of repeaters [41].

Our results in this section are organized as follows. Section 8.1 shows the advantage of irregular topologies over regular ones, such as, mesh in terms of chip area. Section 8.2 notes the communication cost of our PSO based formulation and compares with the approaches [19], [32], and [26], placing routers at corners, centers and intersections of the cores respectively. It also includes a study on the effect of putting constraint on the number of global ports per router. The effect of varying L_{MAX} has also been demonstrated. Section 8.3 shows improvement in energy consumption. Section 8.4 shows the results of ILP formulation on small application core graphs. The results obtained match exactly with the PSO results. The results of [19], [32], and [26] are also shown for these cases.

To start with, we need to generate the floorplan of the corresponding application to act as input to the NoC synthesis problem. Given core dimensions and the communication requirements between the cores, it is desirable to keep the highly communicating cores close to each other. To arrive at such a floorplan, we have first used NMAP [3], a tool to perform application mapping onto a mesh topology. However, it assumes all the cores to be of uniform size. The final mesh floorplan is obtained by aligning cores to the rows, taking care of core sizes.

For example, for the application VOPD, the core graph is shown in Fig. 3a. The corresponding mesh floorplan is noted in Fig. 3b. The mesh floorplan is modified manually to arrive at the custom floorplan in Fig. 3c for the application. Figs. 4 and 5 show the core graph, mesh and custom floorplans for the applications *Auto industry* and *Telecom* respectively. Custom floorplans, so generated, act as input to our router placement problem. It may be noted that the small circles in Figs. 3c, 4c and 5c are the router locations selected by running our tool over the corresponding custom floorplans.

Table 1
Benchmark applications and their core dimensions.

Application	No. of cores	Core dimensions (mm × mm)
PIP	8	2.5 × 1.5, 1 × 1, 1 × 2, 1 × 2.5, 1.5 × 1, 1 × 2.5, 2.5 × 1.5, 1 × 1
VOPD	16	1 × 2.5, 3 × 1, 3 × 3, 2 × 2.5, 2 × 1, 2.5 × 1, 1.5 × 1, 2 × 2.5, 2 × 1.5, 2 × 1.5, 2.5 × 3, 2 × 1.5, 1 × 1, 2 × 1.5, 1 × 1, 2 × 1
MPEG4	12	1.5 × 2, 2 × 1, 2.5 × 1.5, 3 × 2.5, 1 × 3, 2 × 1.5, 2 × 1.5, 1 × 2, 1.5 × 3, 2 × 1.5, 2 × 2.5, 3 × 2.5
MWD	12	2.5 × 1, 2.5 × 2.5, 3 × 2.5, 2.5 × 1.5, 1 × 2.5, 1.5 × 1.5, 2 × 1.5, 2.5 × 2, 2 × 1, 2 × 1.5, 1.5 × 2, 1.5 × 2.5
263 enc mp3 dec	12	2 × 1, 3 × 1, 3 × 2, 1.5 × 2, 1.5 × 3, 1.5 × 3, 2.5 × 2, 2 × 2, 2.5 × 2, 2 × 1.5, 1.5 × 1.5, 2 × 1
263 dec mp3 dec	14	1.5 × 1.5, 1 × 1.5, 1.5 × 1, 2 × 2.5, 3 × 1.5, 2 × 1, 1.5 × 2, 1 × 1.5, 1.5 × 2, 2.5 × 1, 2.5 × 2, 2.5 × 3, 1.5 × 3, 2.5 × 1.5
Mp3 enc mp3 dec	13	1.5 × 2.5, 1.5 × 2, 1.5 × 2.5, 1 × 2.5, 1.5 × 2.5, 1.5 × 2, 1 × 3, 2 × 2, 2.5 × 2.5, 2 × 1, 1 × 2, 2 × 2, 2 × 1.5
Office Automation	5	1 × 3, 3 × 2, 3 × 3, 2.5 × 1, 2 × 3
Consumer	12	3 × 1.5, 1.5 × 2, 3 × 2, 2 × 2, 2 × 1.5, 3 × 1.5, 2.5 × 1, 2 × 1.5, 1.5 × 2, 2 × 2, 1.5 × 1.5, 3 × 1.5
Networking	13	2.5 × 1.5, 1.5 × 1.5, 1.5 × 1.5, 2 × 1, 2 × 1, 2 × 1, 2 × 1.5, 2 × 1.5, 2 × 1.5, 3 × 1.5, 2.5 × 1.5, 3 × 1, 3 × 1
Auto Industry	24	2 × 2, 1.5 × 1.5, 1.5 × 2, 1.5 × 2, 1.5 × 1.5, 1.5 × 2, 1.5 × 1, 1.5 × 1, 1.5 × 2, 2 × 1.5, 2 × 1.5, 1.5 × 2, 2.5 × 1.5, 1.5 × 2, 1.5 × 1.5, 1.5 × 2, 2 × 1.5, 1.5 × 2, 2.5 × 1.5, 2 × 1.5, 2 × 1.5, 1 × 1.5, 1.5 × 2, 1.5 × 1.5
Telecom	30	1.5 × 1, 1.5 × 1.5, 2 × 2, 1 × 1.5, 1.5 × 2, 1.5 × 1.5, 2.5 × 1.5, 2 × 1.5, 2 × 1.5, 2.5 × 1.5, 2 × 1.5, 1.5 × 2, 2 × 1.5, 2 × 1.5, 1.5 × 1.5, 1 × 1.5, 1.5 × 1, 2 × 1, 2 × 1, 1.5 × 1.5, 2 × 1.5, 1 × 1.5, 2 × 1.5, 1.5 × 1, 2 × 2, 2 × 1.5, 1 × 2, 1.5 × 1, 1.5 × 1.5, 1.5 × 1.5

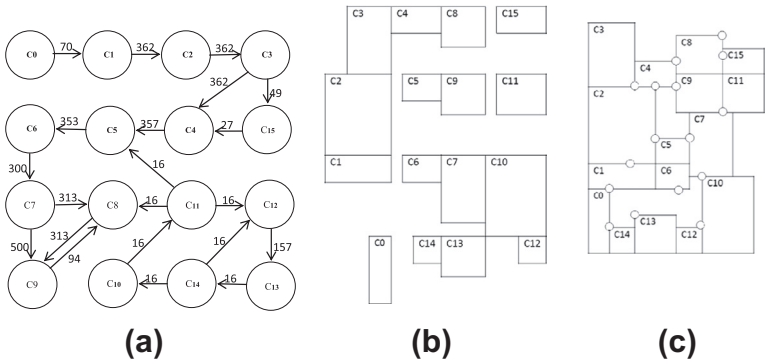

Fig. 3. VOPD. (a) Communication trace graph, (b) mesh floorplan and (c) custom floorplan.

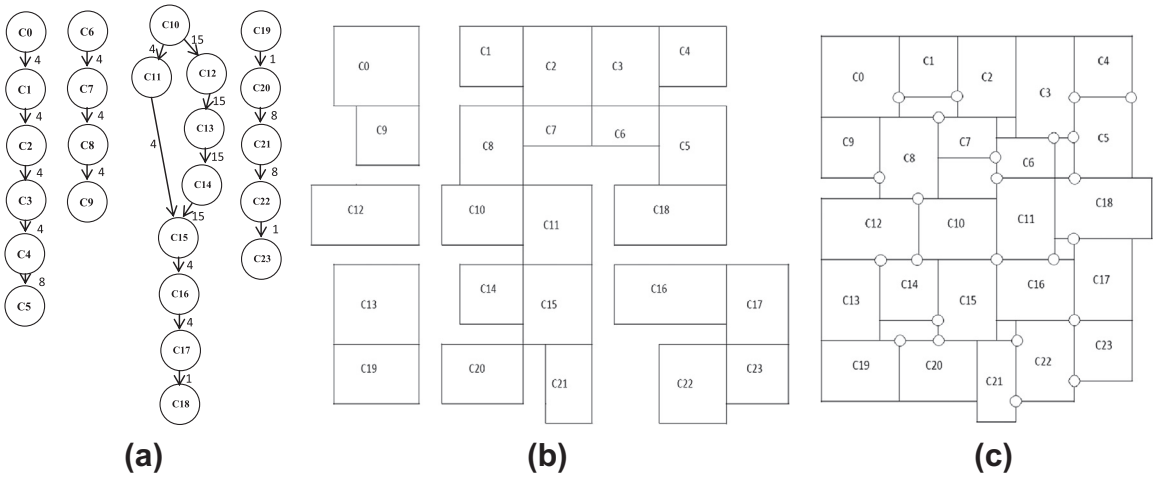

Fig. 4. Auto industry. (a) Communication trace graph, (b) mesh floorplan and (c) custom floorplan.

8.1. Area improvement in irregular topologies

The area is calculated by multiplying the number of grids used in *X*-direction with the number of grids used in *Y*-direction in the floorplan of the applications in mesh and in the irregular topology. Table 2 shows the percentage saving in area in custom floorplan over the mesh based approach. For different applications, it shows upto 45% saving in area, with an average of 36.84% over mesh based floorplan. As the cores of the mesh are aligned to the rows of the grid, a mesh based floorplan takes more area compared to custom floorplan.

8.2. Communication cost results for PSO

Table 3 shows a comparison in communication cost between the PSO based approach proposed in this paper and those in [19], [32], and [26]. All the four techniques have been given the same floorplan as input, the maximum link length L_{MAX} has been set to

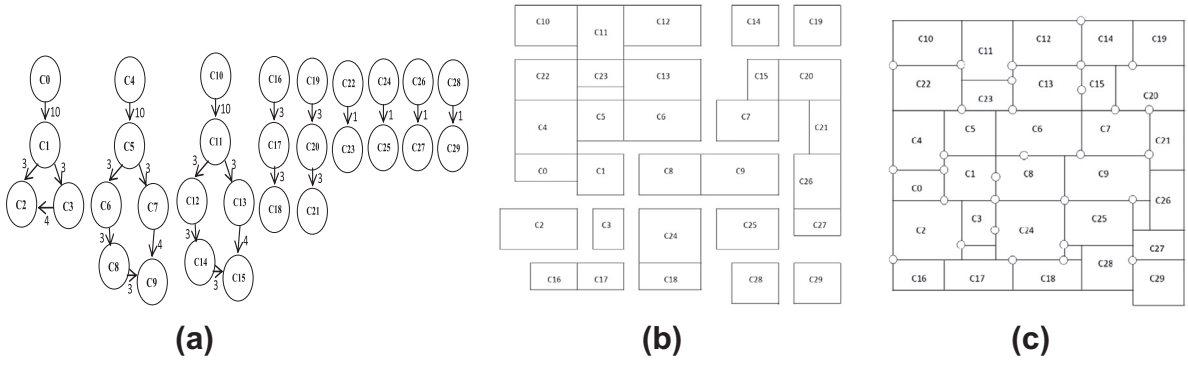

Fig. 5. Telecom. (a) Communication trace graph, (b) mesh floorplan and (c) custom floorplan.

Table 2
Area comparison between mesh based approach and custom floorplan.

Application	Percentage area saving over mesh based approach
PIP	45.09
VOPD	31.1
MPEG4	39.04
MWD	34.37
263 enc mp3 dec	32.29
263 dec mp3 dec	37.95
Mp3 enc mp3 dec	38.09
Office Automation	16.66
Consumer	32.89
Networking	14.28
Auto Industry	23.09
Telecom	36.16

2.5 mm and each router may contain at most four global ports. With this setting, the PSO could identify router locations successfully for all the benchmarks, however, each of the other three techniques fails in many cases (indicated as "NO PATH" in the table). Considering the successful cases also, the approach proposed in [19] needs 5.24 times communication cost than PSO, while [32] needs 4.19 times and [26] needs 1.4 times. Hence, it is demonstrated that the PSO could find better locations for placing the routers so that the communication cost gets reduced as a result of decreasing number of hops in individual communication between cores.

Next, we show a study on the effect of variation in number of global ports (G) at each router over the communication cost. In Table 4, we have noted results for four different cases. The case $G = \infty$ does not put any constraint on the number of global ports, G = 4 allows at most four global ports per router, and so on. The link length constraint L_{MAX} has been kept at 2.5 mm. Compared to unconstrained version ($G = \infty$), the case G = 4 incurs 1.01 times communication cost, while G = 3 and G = 2 incur 1.02 times and 1.19 times costs respectively. As the number of global ports decreases, the number of neighboring routers that a router can connect to, decreases. In effect, the number of hops in the path increases, resulting in an increase in the communication cost.

In Table 5, we show the comparison of communication cost in our approach by varying the link length (L_{MAX}), keeping the number of global ports equal to 4. The results are presented for L_{MAX}=2.5 mm to L_{MAX}=0.75 mm. For link length less than 0.75 mm, we have observed that there exists no path in the router graph for communication between some of the cores present in core graph (indicated as "NO PATH" in the table). As the link length

Table 4
Communication cost comparison with variation of global ports for different applications.

Application	Communication Cost (in no. of hops * bandwidth) Our approach (PSO)7			
	$G = \infty$	G = 4	G = 3	G = 2
PIP	576	576	640	768
VOPD	3811	4076	4672	8175
MPEG4	3466	3499	3772	5054
MWD	1120	1120	1184	1888
263 enc mp3 dec	230.21	230.24	230.48	272.64
263 dec mp3 dec	19.63	19.81	20.14	22.11
Mp3 enc mp3 dec	16.52	16.55	17.07	20.61
Office Automation	2363	2363	2363	2363
Consumer	38,000	38,000	40,000	49,000
Networking	88,080.38	88,080.38	88,080.38	96,468.99
Auto Industry	123	135	178	600
Telecom	92	100	132	460
Average	1.0	1.01	1.02	1.19

Table 3
Communication cost comparison between approaches in [19,32,26] and PSO.

Application	Routers at corners [19]	Routers at centers [32]	Routers at intersections[26]	PSO
PIP	896	3584	896	576
VOPD	NO PATH	NO PATH	NO PATH	3811
MPEG4	NO PATH	NO PATH	5213	3466
MWD	NO PATH	NO PATH	NO PATH	1120
263 enc mp3 dec	NO PATH	NO PATH	NO PATH	230.21
263 dec mp3 dec	NO PATH	NO PATH	NO PATH	19.64
Mp3 enc mp3 dec	NO PATH	NO PATH	NO PATH	16.52
Office Automation	NO PATH	NO PATH	NO PATH	2363
Consumer	94000	NO PATH	58000	38000
Networking	205520.905	188743.68	NO PATH	88080.38
Auto Industry	558	351	NO PATH	135
Telecom	226	247	317	100
Average	5.24	4.19	1.4	1.0

Table 5
Communication cost comparison with variation of link length for different applications.

Application	Our approach (PSO)				
	L_{MAX} = 2.5 mm	L_{MAX} = 2 mm	L_{MAX} = 1.5 mm	L_{MAX} = 1 mm	L_{MAX} = 0.75 mm
PIP	576	576	640	1216	NO PATH
VOPD	4016	8513	NO PATH	NO PATH	NO PATH
MPEG4	3499	3500.5	5790.5	NO PATH	NO PATH
MWD	1120	1184	1536	NO PATH	NO PATH
263 enc mp3 dec	230.24	233.53	421.36	NO PATH	NO PATH
263 dec mp3 dec	19.81	20.39	31.17	NO PATH	NO PATH
Mp3 enc mp3 dec	16.59	18.09	22.88	NO PATH	NO PATH
Office Automation	2363	2363	2363	2363	2364
Consumer	38,000	38,000	40,000	NO PATH	NO PATH
Networking	88,080.38	88,080.38	88,080.38	NO PATH	NO PATH
Auto Industry	135	176	NO PATH	NO PATH	NO PATH
Telecom	100	134	NO PATH	NO PATH	NO PATH

Table 6
Power consumption values of different port routers.

Router ports	Power (mW)
2	17.1425
3	24.79
4	42.1068
5	51.5325

Table 7
Network energy comparison between mesh and PSO.

Application	Percentage network energy saving over Mesh
PIP	23.44
VOPD	15.17
MPEG4	22.60
MWD	35.76
263 enc mp3 dec	30.11
263 dec mp3 dec	30.48
Mp3 enc mp3 dec	29.11
Office Automation	36.85
Consumer	33.01
Networking	51.87
Auto Industry	44.39
Telecom	42.37

decreases, the communication cost gets increased, since, the connections between the routers decrease as the distance between the routers does not satisfy the link length constraint.

8.3. Energy consumption

Though in our optimization process, we have reduced the communication cost, it is customary to check its effect on the overall network energy consumption. Network energy consists of two components – energy consumed by the routers and the energy consumed by the links. Again, the routers may have different number of ports. Router complexity depends on the number of global ports it has. To get an estimate of the router energy/power, we have synthesized virtual-channeled wormhole routers with different number of ports using Synopsys Design Vision tool [42] in 90 nm CMOS technology. The details of the router design can be found in [43]. The routers can be operated at a minimum frequency of 1.5 GHz. Synopsys Prime Power tool [44] has been utilized to determine the power consumption in each router. Table 6 shows the power consumption values. Link energy is obtained through Cadence HSPICE [45]. For a link length of 2.5 mm, the value obtained is 0.499625 μJ.

Table 7 shows the percentage saving in network energy in custom floorplan (using PSO) over mesh based approach. For different applications, it shows up to 51% saving in network energy, with an average of 32.93% over mesh based floorplan.

Table 8
Communication cost comparison between ILP, PSO and approaches [19,32,26].

Application (No. of cores)		Office Automation(5)	G1(5)	G2(4)	G3(5)
L_{MAX} = 2.5 mm	ILP	**2363**	**403.04**	**539.78**	**746.55**
	PSO	**2363**	**403.04**	**539.78**	**746.55**
	Routers at corners [19]	NO PATH	**403.04**	1493.10	1251.33
	Routers at centers [32]	NO PATH	NO PATH	NO PATH	907.78
	Routers at intersections [26]	NO PATH	**403.04**	1161.35	1251.33
L_{MAX} = 3 mm	Routers at corners [19]	NO PATH	**403.04**	699.75	1078.30
	Routers at centers [32]	NO PATH	**403.04**	1493.10	907.78
	Routers at intersections [26]	3937	**403.04**	699.75	1161.35
L_{MAX} = 4 mm	Routers at corners [19]	2365	**403.04**	**539.78**	**746.55**
	Routers at centers [32]	2365	**403.04**	699.75	**746.55**
	Routers at intersections [26]	3150	**403.04**	**539.78**	**746.55**
L_{MAX} = 4.5 mm	Routers at corners [19]	**2363**	**403.04**	**539.78**	**746.55**
	Routers at centers [32]	**2363**	**403.04**	**539.78**	**746.55**
	Routers at intersections [26]	**2363**	**403.04**	**539.78**	**746.55**

8.4. ILP Results

In this section, we present the results of our ILP formulation and compare them with PSO and the works [19], [32], [26] reported in the literature. ILP produces optimum results, however takes a good amount of CPU time for the same. We have used the ILP results to validate the PSO approach. For this, we have worked with a few small applications having up to five cores. One benchmark, *Office*

Table 9
CPU time comparison between ILP and PSO.

Application	Number of cores	ILP CPU Time(s)	Our approach (PSO) CPU time (μs)
Office Automation	5	3584.44	95
G1	5	3314.46	53
G2	4	2649.61	3
G3	5	2989.51	10

Automation satisfies this requirement. Other three application graphs G1, G2 and G3 have been generated using the TGFF tool [46]. Table 8 summarizes the results. For ILP, we have used CPLEX [36], a commercial ILP solver to solve the formulated ILP. Putting maximum allowed link length L_{MAX} at 2.5 mm, the results obtained by PSO match exactly with those from ILP. However, other tools ([19], [32], [26]) either fail to find all communication paths through the network, or produce highly suboptimal results. To determine the minimum link length at which those tools can also produce optimum results, we have extended L_{MAX} to 3 mm, 4 mm and 4.5 mm. Allowing larger links could get better results for them, however, the link energy consumption will go up, increasing the network energy. Table 9 shows the timing comparison between ILP and PSO. Both ILP and PSO have been implemented on a PC with Intel dual core processor, operating at 2 GHz and having 3 GB main memory. As it can be observed, PSO could produce results similar to ILP at a much lower CPU time.

9. Conclusion

In this paper, we have demonstrated that the proper choice of router locations within the floorplan of a NoC has a profound effect on the communication overhead, particularly in the presence of maximum link length constraint and restriction on the number of global ports per router. The problem has been solved using an ILP based exact method and a PSO based meta-search strategy. It has been shown that the proposed router placement strategy could result in less communication overhead compared to the techniques reported in the literature. It has been shown that there is significant improvement in network energy in our approach over regular NoC architectures. The future work includes designing a proper scheduling policy for the whole NoC, coming up with floorplanning techniques, taking care of both power and thermal issues.

References

[1] W.J. Dally, B. Towles, Route packets, not wires: on-chip interconnection networks, in: Proceedings of the Design Automation Conference, 2001, pp. 684–689.
[2] Jingcao Hu, R. Marculescu, Energy-aware mapping for tile-based NoC architectures under performance constraints, in: Proceedings of the ASP-DAC 2003. Asia and South Pacific Design Automation Conference, 21–24 January, 2003, pp. 233–239.
[3] S. Murali, G. De Micheli, Bandwidth constrained mapping of cores onto NoC architectures, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition, vol. 2, February 2004, pp. 896–901.
[4] G. Ascia, V. Catania, M. Palesi, Multi-objective mapping for mesh-based NoC architectures, in: CODES + ISSS 2004. International Conference on Hardware/Software Codesign and System Synthesis, 8–10 Sept. 2004, pp. 182–187.
[5] L. Benini, Application Specific NoC Design, in: Proceedings of the Design, Automation and Test in Europe, vol. 1, 6–10 March, 2006, pp. 1–5.
[6] W.E. Donath, Complexity theory and design automation, in: Proceedings of the 17th Conference on Design Automation, 23–25 June 1980, pp. 412–419.
[7] U.Y. Ogras, R. Marculescu, Application-specific network-on-chip architecture customization via long-range link insertion, in: ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 6–10 November, 2005, pp. 246–253.
[8] A. Pinto, L.P. Carloni, A.L. Sangiovanni-Vincentelli, Efficient synthesis of networks on chip, in: Proceedings of the 21st International Conference on Computer Design, 13–15 October, 2003, pp. 146–150.
[9] I. Korotkyi, O. Lysenko, Application-specific network-on-chip with link aggregation, in: 2012 Mediterranean Conference on Embedded Computing (MECO), 19–21 June, 2012, pp. 9–12.
[10] A.A. Morgan, H. Elmiligi, M.W. El-Kharashi, F. Gebali, Area-aware topology generation for Application-Specific Networks-on-Chip using network partitioning, in: PacRim 2009. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 23–26 August, 2009, pp. 979–984.
[11] A. M'zah, O. Hammami, Area/delay driven NoC synthesis, in: 2011 International Conference on Microelectronics (ICM), 19–22 December, 2011, pp. 1–61.
[12] A.A. Morgan, H. Elmiligi, M. Watheq El-Kharashi, F. Gebali, Networks-on-Chip topology generation techniques: area and delay evaluation, in: IDT 2008. Third International Conference on Design and Test Workshop, 20–22 December, 2008, pp. 33–38.
[13] Sami J. Habib, Mohammad Gh. Mohammad, Q8WARE: synthesis tool for Network-on-Chip applications, in: Innovations in Information Technology, November 2006, pp. 1–5.
[14] K. Srinivasan, K.S. Chatha, SAGA: synthesis technique for guaranteed throughput NoC architectures, in: Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, vol. 1, 18–21 January, 2005, pp. 489–494.
[15] I. Filippopoulos, I. Anagnostopoulos, A. Bartzas, D. Soudris, G. Economakos, Systematic exploration of energy-efficient Application-Specific Network-on-Chip architectures, in: 2010 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 5–7 July, 2010, pp. 133–138.
[16] V. Dumitriu, G.N. Khan, Throughput-oriented NoC topology generation and analysis for high performance SoCs, in: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 17, no. 10, October 2009, pp.1433–1446.
[17] Y. Ar, S. Tosun, H. Kaplan, TopGen: a new algorithm for automatic topology generation for Network on Chip architectures to reduce power consumption, in: AICT 2009. International Conference on Application of Information and Communication Technologies, 14–16 October, 2009, pp. 1–5.
[18] A. Jalabert, S. Murali, L. Benini, G. De Micheli, ×pipesCompiler: a tool for instantiating application specific networks on chip, in: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, vol. 2, 16–20 February, 2004, pp. 884–889.
[19] M. Reza Kakoee, F. Angiolin, S. Murali, A. Pullini, C. Seiculescu, L. Benini, A floorplan-aware interactive tool flow for NoC design and synthesis, in: SOCC 2009. Proceedings of IEEE International SOC Conference, 9–11 September, 2009, pp. 379–382.
[20] G. Leary, K. Srinivasan, K. Mehta, K.S. Chatha, Design of Network-on-Chip Architectures with a genetic algorithm-based technique, in: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 17, no. 5, May 2009, pp. 674–687.
[21] U.Y. Ogras, R. Marculescu, Energy- and performance-driven NoC communication architecture synthesis using a decomposition approach, in: Proceedings of the Design, Automation and Test in Europe, 7–11 March, 2005, pp. 352–357.
[22] Guoming Lai, Xiaola Lin, Siyan Lai, GA-based floorplan-aware topology synthesis of application-specific network-on-chip, in: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS), vol. 2, 29–31 October, 2010, pp. 554–558.
[23] A. Tino, G.N. Khan, Multi-objective Tabu Search based topology generation technique for application-specific Network-on-Chip architectures, in: Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 14–18 March, 2011, pp. 1–6.
[24] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, G. De Micheli, ×pipes Lite: a synthesis oriented design library for networks on chips, in: Proceedings of the Design, Automation and Test in Europe, 7–11 March, 2005, pp. 1188–1193.
[25] T. Ahoen, A. David, H. Bin, J. Nurmi, Topology optimization for application specific networks on chip, in: Proceedings of the International Workshop on System level Interconnect Prediction, February 2004, pp. 53–60.
[26] K. Srinivasan, K.S. Chatha, G. Konjevod, An automated technique for topology and route generation of application specific on-chip interconnection networks, in: ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 6–10 November, 2005, pp. 231–237.
[27] K. Srinivasan, K.S. Chatha, G. Konjevod, Linear-programming-based techniques for synthesis of network-on-chip architectures, in: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, no. 4, April, 2006, pp. 407–420.
[28] S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. De Micheli, L. Raffo, Designing Application-Specific Networks on Chips with Floorplan Information, in: ICCAD '06. IEEE/ACM International Conference on Computer-Aided Design, 5–9 November, 2006, pp. 355–362.
[29] J. Chan, S. Parameswaran, NoCOUT: NoC topology generation with mixed packet-switched and point-to-point networks, in: ASPDAC 2008. Asia and South Pacific Design Automation Conference, 21–24 March, 2008, pp. 265–270.
[30] Bei Yu, Sheqin Dong, Song Chen, S. Goto, Floorplanning and topology generation for application-specific Network-on-Chip, in: 2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC), 18–21 January, 2010, pp. 535–540.
[31] Shan Yan, B. Lin, Application-specific Network-on-Chip architecture synthesis based on set partitions and Steiner Trees, in: ASPDAC 2008. Asia and South Pacific Design Automation Conference, 21–24 March, 2008, pp. 277–282.

[32] Wei Zhong, Bei Yu, Song Chen, T. Yoshimura, Sheqin Dong, S. Goto, Application-specific Network-on-Chip synthesis: cluster generation and network component insertion, in: 12th International Symposium on Quality Electronic Design (ISQED), 14–16 March, 2011, pp. 1–6.
[33] D. Bertozzi, A. Jalabert, Srinivasan Murali, R. Tamhankar, S. Stergiou, L. Benini, G. De Micheli, NoC synthesis flow for customized domain specific multiprocessor systems-on-chip, in: IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 2, February 2005, , pp. 113–129.
[34] Soohyun Kwon, S. Pasricha, Jeonghun Cho, POSEIDON: a framework for application-specific Network-on-Chip synthesis for heterogeneous chip multiprocessors, in: 12th International Symposium on Quality Electronic Design (ISQED), 14–16 March, 2011, pp. 1–7.
[35] G.N. Khan, A. Tino, Synthesis of NoC Interconnects for Custom MPSoC Architectures, in: Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS), 9–11 May, 2012, pp. 75–82.
[36] www.ibm.com/software/in/integration/optimization/cplex.
[37] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings., IEEE International Conference on Neural Networks, vol. 4, Nov/Dec 1995, pp. 1942–1948.
[38] Kang-Ping Wang, Lan Huang, Chun-Guang Zhou, Wei Pang, Particle swarm optimization for traveling salesman problem, in: International Conference on Machine Learning and Cybernetics, vol. 3, 2–5 November, 2003, pp. 1583–1585.
[39] Luo Guilan, Zhao Hai, Song Chunhe, Convergence analysis of a dynamic discrete PSO algorithm, in: ICINIS '08. First International Conference on Intelligent Networks and Intelligent Systems, 1–3 November, 2008, pp. 89–92.
[40] E. Salminen, A. Kulmala, T.D. Hmlinen, Survey of network-on-chip proposals, Tech. Rep, March 2008 [Online]. Available from: <www.ocpip.org/socket/whitepapers>.
[41] R. Ho, K.W. Mai, M.A. Horowitz, The future of wires, Proceedings of the IEEE 89 (4) (Apr 2001) 490–504.
[42] Design Vision User Guide, Version U-2003.03, March 2003, Synopsys, Inc.
[43] Santanu Kundu, J. Soumya, Santanu Chattopadhyay, Design and evaluation of Mesh-of-Tree based Network-on Chip using virtual channel router, Microprocessors and Microsystems 36 (6) (2012) 471–488.
[44] Primepower Manual, Version Y-2006.06, June 2006, Synopys, Inc.
[45] HSPICE Reference Guide, U-2003.09, September 2003.
[46] R.P. Dick, D.L. Rhodes, W. Wolf, TGFF: task graphs for free, in: CODES/CASHE '98, Proceedings of the Sixth International Workshop on Hardware/Software Codesign, 15–18 March 1998, pp. 97–101.

Soumya J. received B.Tech degree in Electronics and Communications Engineering from JNTU, Hyderabad and Masters degree in Embedded Systems from Indian Institute of Technology Kharagpur. She is currently working towards Ph.D. degree in Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur. Her research interests include Application Specific Network on Chip and Reconfigurable Network on Chip design.

Santanu Chattopadhyay did his B.E. in Computer Science and Technology from University of Calcutta (Bengal Engineering College) in1990.He completed his M.Tech in Computer and Information Technology and Ph.D. in Computer Science and Engineering, in 1992 and 1996 respectively, both from Indian Institute of Technology Kharagpur, India. He is currently a Professor in the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, India. His research interests include low power circuit design and test, System-on-Chip test, Network-on-Chip design and test. He has published more than150 technical papers in refereed international journals and conferences.