Application Mapping onto Butterfly-Fat-Tree based Network-on-Chip using Discrete Particle Swarm Optimization

Pradip Kumar Sahu, Kanchan Manna and Santanu Chattopadhyay
Electronics and Electrical Communication Engineering,
Indian Institute of Technology, Kharagpur, West Bengal, India

ABSTRACT

This paper addresses the problem of application mapping onto Butterfly-Fat-Tree (BFT) based Network-on-Chip design. It proposes a new mapping technique based on discrete Particle Swarm Optimization (PSO) to map the cores of the core graph to the routers. The basic PSO has been augmented by running multiple PSO and deterministically generating a part of the initial population for PSO. The mapping results have been compared with well-known techniques reported in the literature for a number of benchmark applications. The reported strategy produces results superior to those obtained via existing approaches within a reasonable CPU time.

Keywords

Application mapping, Network-on-Chip, System-on-Chip, Butterfly-Fat-Tree, Discrete Particle Swarm Optimization

1. INTRODUCTION

Network-on-Chip (NoC) has evolved as a viable strategy to implement Intellectual Property (IP) core based System-on-Chip (SoC) designs. It solves the traditional problems of bandwidth limitations of bus-based SoC design by providing an on-chip network fabric consisting of routers connected in a certain topology. Conventional data signal exchanges are replaced by message passing between the cores through the router fabric [1]–[4]. A major challenge in NoC based system design is to associate the IP cores implementing tasks of an application with routers. This is commonly known as the process of application mapping. This has got a very significant role to play in performance of the overall system as it directly influences the communication time, the required link bandwidth, the admissible delay of the router, and energy consumption of the whole NoC [5], [6]. Furthermore, these requirements vary from one application domain to another. For example, while multimedia applications require high bandwidth, real time systems require guaranteed delay, and portable devices require low power consumption. Most of the application mapping algorithms reported in the literature assumes a mesh-connected router fabric for the NoC. In [6]–[10], authors have proposed many topologies for NoC. Butterfly-Fat-Tree (BFT) enjoys several advantages over other topologies [7], [10]. The BFT topology can be easily implemented inside chips. It has the advantage of having small diameter as well as symmetric structure. There are various versions of the networks connected in the tree like architecture and most of them have recursive structures. The wire routing is also simpler. In chip design, for same number of cores, the area occupied by BFT is less than the mesh topology. These characteristics make BFT a popular scheme for interconnecting IPs [7], [10]. In [7], [9]–[10] authors have shown a detailed performance comparison of BFT with mesh using synthetic traffic. However, it remains unclear how BFT performs with respect to the benchmark applications. This work attempts to map the benchmark application onto BFT based NoC. Such process can be called as *application mapping*.

Application mapping problem is NP-hard [11]. This paper explores a meta-heuristic, *Discrete Particle Swarm Optimization* (*DPSO*), to perform application mapping onto BFT-based NoCs. The salient features of the approach are as follows.

1. A *PSO* based approach has been presented for application mapping onto BFT targeting minimization of the overall communication cost.
2. We have also used a multi-stage *PSO*. The local and global best information of i^{th} stage have been passed to the particles in $(i+1)^{th}$ stage. This ensures faster convergence and improved quality of solution for the successive stages.
3. For any stage of *PSO*, the initial population generation is not fully random. A good number of particles have been created using a heuristic. This has enabled our *PSO* to explore the promising regions of search space much better.
4. The communication cost metric values of the mapping solutions of our approach have been compared with the existing approach for BFT mapping. It shows good improvement in solution quality.
5. Comparison of dynamic performance (in terms of average network latency) and energy consumption have also been carried out.

The rest of the paper is organized as follows. Section 2 gives a brief survey of previous works on application mapping. Section 3 gives an overview of BFT. Section 4 presents the problem formulation. *DPSO* formulation of the mapping problem and its augmentations are presented in Section 5. Section 6 embodies performance analysis of our approach and compares with others by taking some real SoC benchmarks. Latency and energy values have also been compared. Section 7 draws the conclusion.

2. RELATED WORKS

Application mapping onto mesh structured NoC is a well studied area. A recent survey on this can be found in [12]. PMAP, a two-phase mapping algorithm for placing clusters onto processors has been presented in [11], where highly communicating clusters are placed on adjacent nodes of the processor network. Each cluster contains all tasks which are to be executed in the same processor having zero interconnection overhead to increase parallelism. In [13], GMAP, and PBB a branch and bound algorithm, have been proposed that map cores onto a tile-based NoC architecture satisfying the bandwidth constraint and minimizing the total energy consumption. In [14], NMAP, a mapping technique has been proposed with minimum path routing in the mesh architecture which satisfies the bandwidth constraint and minimizes the average communication delay. MOCA, a two phase heuristic for low energy mesh based on-chip interconnection architecture has been proposed in [15]. In the first phase, the cores are mapped to different routers of the mesh by invoking a bi-partitioning based slicing tree generation technique. In the second phase, it attempts to find a minimal path from source to destination for each traffic trace. A binomial IP mapping and optimization algorithm (BMAP) has been proposed in [16] to reduce hardware cost of on-chip network. Spiral, a mapping algorithm has been proposed in [17] which reduce the cumulative energy consumption of communication links and the overall system execution time. In this mapping technique, the high priority resources are mapped spirally from the centre to the boundaries of the mesh based NoC by placing highly communicating cores as close as possible to each other. Onyx, a bandwidth constrained application mapping has been presented in [18] to minimize the overall communication cost of NoC. CHMAP [19] is a chain-mapping algorithm that produces chains of connected cores in order to introduce a method for application mapping onto mesh-based NoC. CMAP [20] is a constructive application mapping algorithm that maps cores onto NoC minimizing total communication cost and energy. In [21], authors have taken NMAP [14] as their initial mapping solution. A branch-and-bound algorithm, as in [13], has been applied upon the NMAP mapping solution to arrive at a better solution. CastNet, an energy-aware application mapping and routing technique for NoC has been proposed in [22]. In [23]–[24], a mapping algorithm has been proposed to reduce both static and dynamic cost of mesh based NoC using Kernighan-Lin (*K-L*) partitioning scheme. A thermal uniformity-aware application mapping strategy onto mesh-based NoC has been proposed in [25] using a constructive heuristic to make the temperature profile uniform, as well as reduce the peak temperature with tolerable performance degradation.

A two-step Genetic Algorithm (*GA*) for mapping applications onto NoC has been proposed in [26], which reduces the overall execution time. In the first step, the cores of a core graph are assigned onto different IPs assuming the edge delays to be constant and equal to the average edge delay. In second step, the IPs are mapped to tiles of NoC taking the actual edge delay based on the network traffic model, and the total system delay is minimized. A multi-objective Genetic Algorithm (MOGA) based application mapping technique has been proposed in [27], where one-one as well as many-many mapping between switches and tiles have been taken into consideration to minimize energy consumption and required link bandwidth. In [28], CGMAP, a genetic algorithm based application mapping technique has been proposed that uses the chaotic mapping operator instead of the random processes in *GA*. GAMR [29], a genetic algorithm based mapping and routing approach addresses a two phase mapping of IP cores onto NoC architecture and generates a deterministic dead-lock free minimal routing path for each communication to minimize the total communication energy and maximum link bandwidth of the NoC architecture. GBMAP, an evolutionary approach for mapping cores onto NoC architecture has been proposed in [30], which reduces energy consumption and total bandwidth requirement of NoC. PLBMR, a Particle Swarm Optimization (*PSO*) based two-phase application mapping algorithm proposed in [31] minimizes the NoC communication energy and allocates the routing path for balancing the link-load. In first phase, the *PSO* maps IP cores onto NoC to minimize the energy consumption, and in the second phase the routing paths are allocated to every pair to satisfy the link-load balance. A mapping technique based on discrete *PSO* has been presented in [32]. However, it only considers improvement over genetic algorithm based method and reports relative improvements only. In [33], a hybrid multi-objective algorithm has been proposed, where Dijkstra shortest path algorithm has been used to find the shortest path among communicating cores to satisfy the bandwidth constraints and then a multi-objective pareto based *PSO* technique is applied upon that to improve performance. PSMAP, a meta-heuristic strategy using *PSO* technique has been proposed in [34] to reduce both static and dynamic cost of NoC for mesh based application mapping. A discrete multiple *PSO* based mapping technique has been proposed [35] to optimize the performances using deterministic initial solutions. In [36], an Ant Colony Optimization (*ACO*) based algorithm has been proposed for application mapping onto NoC to minimize the bandwidth requirement. The results have been compared with random mapping techniques.

All the application mapping techniques of NoC discussed above are based on mesh based network architecture. But it is essential to check the suitability of these approaches in other network topologies when applications are mapped onto that. In this light, application mapping techniques have been proposed in [23], [37] and [23], [38], [39] to map applications onto Butterfly-Fat-Tree (BFT) and Mesh-of-Tree (MoT) based NoC respectively to enhance communication cost.

3. BFT-AN OVERVIEW

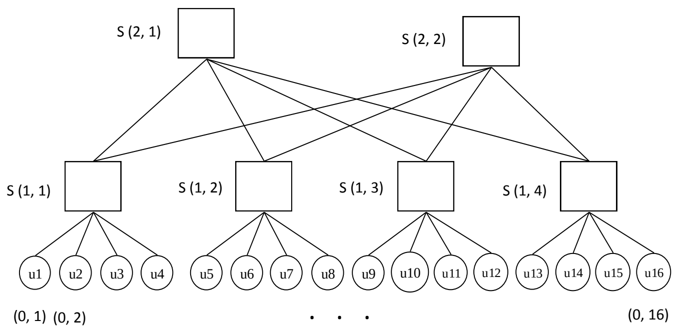


Fig. 1. Butterfly-Fat-Tree topology

Fig. 1 shows a BFT [7, 40] structure in which routers and their interconnections are shown. The IPs are placed at the leaf level. A pair of coordinates (l,p) is used to label each node, where l denotes a node's level and p denotes its position within that level. In general, at the lowest level, there are N functional IPs with addresses ranging from 1 to N. A pair $(0, n)$ denotes the location of an IP at the lowest level. Each switch, denoted by $S\ (l, p)$ has four children ports and two parent ports. The IPs are connected to $N/4$ switches at the first level. For N number of IPs, the diameter of network is $\log_2 N/4$ and the bisection width is $\sqrt{N}$. Fig. 1 shows BFT architecture connecting *16* cores to the routers. The cores are connected at leaf level routers, and each leaf level router connects *4* cores. It has small diameter as well as symmetric structure.

4. PROBLEM FORMULATION

An application consists of a set of tasks, each of which is implemented by an IP core. After the cores participating in an application have been decided, the application can be represented in the form of a *core graph* [14], defined as follows.

Definition 1: The *core graph* for an application is a directed graph, $G(C,E)$ with each vertex $c_i \in C$ representing a core and the directed edge $e_{i,j} \in E$ representing the communication between the cores c_i and c_j . The weight of edge $e_{i,j}$, denoted by $comm_{i,j}$, represents the bandwidth requirement of the communication from c_i to c_j .

Definition 2: The NoC *topology graph* [14] is a directed graph $P(U,F)$ with each vertex $u_i \in U$ representing a node in the topology and the directed edge $f_{i,j} \in F$ representing a direct communication between the vertices u_i and u_j . The weight of the edge $f_{i,j}$, denoted as $bw_{i,j}$, represents the bandwidth available across the edge $f_{i,j}$.

A mapping of the core graph $G(C,E)$ onto the topology graph $P(U,F)$ is defined by the function $map: C \rightarrow U, such\ that,\ \forall\ c_i \in C, \exists\ u_j \in U\ and\ map(c_i) = u_j$. The function associates core c_i to node u_j. Naturally, mapping is defined only when $|C| \leq |U|$. The quality of such a mapping is defined in terms of the total $communication\,cost$ of the application under this mapping [14]. So, the $communication\,cost$ between two vertices u_i and u_j is given by the product of bandwidth requirement and number of hops. However, this cost model holds for the case in which the routers are all of equal latency (as in mesh). We have used the BFT router design reported in [10]. For BFT, root routers show a latency of one cycle, if it is a simple FIFO, otherwise it takes two cycles. The stem and leaf routers always show a latency of two cycles. We have converted the hops into the router clock cycles needed by the path. Let us call the path from u_i to u_j as path k. The communication cost of the path is given by,

$$x^k = rc_{i,j} \times bw_{i,j}$$

where, $rc_{i,j}$ = number of router cycles required in the path from u_i to u_j

$bw_{i,j}$ = bandwidth requirement of the path u_i to u_j

Total communication cost of the network is given by,

$$T_x = \sum_{k=1}^{|E|} x^k$$

where $|E|$ is the total number of paths available in the network from different source to destination vertices of the mapped core graph.

5. MAPPING USING DISCRETE PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (*PSO*) [41] 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 (evolves) 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 a *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. A well-known problem that has been attempted to be solved using discrete *PSO* (*DPSO*) technique is the Travelling Salesman Problem (TSP) [42]. This has motivated us to look for a *DPSO* formulation of the application mapping problem.

5.1 Particle Structure

In application mapping problem, *PSO* formulation of the particle corresponds to a possible mapping of cores to the nodes. An example of a particle structure has been shown in Fig. 2. The numbers shown within circles in the boxes are the core numbers present in the core graph. The numbers outside the box are the core positions (nodes) of the topology graph. It is assumed that the nodes are numbered in an increasing order from the left to right position. The figure shows that core 1 is attached to the leaf router 1, called node1; core 4 is placed at node 2, and so on. If the number of nodes present in the topology graph is greater than the number of cores present in the core graph, dummy nodes are added to the core graph to make the two numbers same. Dummy nodes are connected to all core nodes and between themselves. Edges connecting a core node to dummy nodes and the edges between dummy nodes are assigned a cost *zero*. Let N be the number of cores present in the core graph for mapping of cores onto the topology graph, after connecting dummy nodes, if required. For these N cores, there are N node positions in the topology graph. A particle is a permutation of numbers from 1 to N, which shows the placement of cores to the node positions of the topology graph. The overall communication cost is influenced by the position of cores in a particle. In our formulation, the overall communication cost forms the fitness function. Fitness of a particle p_i is equal to the overall communication cost after placement of cores of the core graph to different nodes, as specified by the particle.

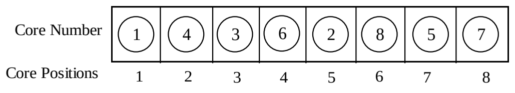


Fig. 2. Particle structure

5.2 Evolution of Generations

In the general *DPSO* framework, let the position of a particle (in an *n*-dimensional space) at k^{th} iteration be $p_k = < p_{k,1}, p_{k,2}, \cdots\ p_{k,n} >$. 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 and $gbest_k$ be the global best particle of iteration k. The new position of particle i is calculated as follows:

$$p_{k+1}^i = (s_1 * I \oplus s_2(p_k \longrightarrow pbest^i) \oplus s_3(p_k \longrightarrow gbest_k)) \cdot p_k^i$$

In the above expressions, $a \longrightarrow b$ represents a 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* >, $a \longrightarrow b$ = < *swap* (*1*, *4*), *swap* (*2*, *4*), *swap* (*3*, *4*) >. The operator $\oplus$ is the fusion operator. For two swap sequences *a* and *b*, $a \oplus b$ is equal to the sequence in which the sequence of swaps in *a* is followed by the sequence of swaps in *b*. The constants s_1, s_2, s_3 are the inertia, self-confidence and swarm confidence values. The quantity $s_i * (a \longrightarrow b)$ means that the swaps in the sequence $a \longrightarrow b$ will be applied with a probability s_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 $s_1 * I \oplus$

$s_2 * (p_k \longrightarrow pbest^i) \oplus s_3 * (p_k \longrightarrow gbest_k)$ is applied on particle p_k^i to generate p_{k+1}^i.

In reference to the application mapping problem, for a particle p, the node associated with a core is identified by the position index of the core in p. The indexing of the position takes value between 1 and N(N being the number of nodes). The index corresponds to the core positions in the topology graph (Fig. 2). Let the swap operator $SO_{j,k}$ ($where, j\ and\ k = 1, 2, \cdots N$) be such that it swaps j^{th} and k^{th} positions of particle p to create a new particle p_{new}. For example, let us consider the particle $p = \{1, 4, 3, 6, 2, 8, 5, 7\}$, where the numbers represent the core numbers of the core graph and the position represents the core positions in the topology graph. The swap operator $SO_{4,6}$ swaps the cores at position 4 and 6, which creates a new particle $p_{new} = \{1, 4, 3, 6, 5, 8, 2, 7\}$.

To align a particle p_i with its local best, the swap sequence is identified. Let this be $SS_i^{l\_best}$. Then another swap sequence is identified to align the particle with the global best. Let this be $SS_i^{g\_best}$. Now the swap sequence $SS_i^{l\_best}$ is applied on particle p_i with a probability of s_2 [43]. Let the modified particle be $p_i^{l\_best}$. Then the swap sequence $SS_i^{g\_best}$ is applied on $p_i^{l\_best}$ with a probability of s_3[43]. This creates a new particle p_i^{new}. Its fitness is evaluated and the local best is updated for particle i, if it is better than the previous local best for the particle. If the best fitness in a generation is better than the global best of the previous generation, the global best is also updated. The procedure $Compute\_Swap\_Sequence$ shows the strategy to determine a swap sequence for the purpose of alignment.

Procedure Compute_Swap_Sequence

```
Input: Source sequence $Sour\_seq$, Destination sequence $Dest\_seq$
Output: Swap sequence $Swap\_seq$ to align $Sour\_seq$ to $Dest\_seq$
Begin
    For $i = 1$ to $length\ (Sour\_seq)$
        $Swap\_seq[i]$ = Index of $Sour\_seq[i]$ in $Dest\_seq$
    End For
End
```

5.3 Convergence of DPSO

From [44], it can be found that the convergence condition for this *DPSO* is given by,

$$(1-\sqrt{s_1})^2 \leq s_2+s_3 \leq (1+\sqrt{s_1})^2$$

Accordingly, we have worked with various values of $s_1, s_2\ and\ s_3$ in this range. The values $s_1 = 1.0$, $s_2 = 0.04, s_3 = 0.02$ have been observed to produce good results for most of the applications we have experimented with.

5.4 PSO Algorithm

```
Initialization:

For each particle
        Initialize particle with random solution
        Evaluate $fitness$ value of each particle
        Set $local\_best$ of each particle to itself
End For
Set $global\_best$ to the best fit particle

Evolution:
Do
    For each particle $p_i$
        Identify $SS_i^{l\_best}$ and $SS_i^{g\_best}$
        $p_i^{new}$ = Modify $p_i$ by applying $SS_i^{l\_best}$
                with probability $s_2$ followed by $SS_i^{g\_best}$
                with probability $s_3$
        Evaluate $fitness$ of $p_i^{new}$

        If $fitness$ of $p_i^{new}$ is better than the local best
                for $p_i$ then update $local\_best$ for $p_i$
    End For
    Find the particle with the best $fitness$ and
            update $global\_best$
While maximum generation (pre-specified) not attained and
$global\_best$ is not remaining unaltered for a pre-specified
number of generations
```

5.5 Augmentations to the PSO

The DPSO formulation discussed above has been augmented in the following two ways to achieve better solutions.

1. *Multiple PSO*: Similar to any other search procedure, PSO also performs both exploration and exploitation of the search space. The exploration process explores different regions of the search space, while the exploitation process checks for local minima around the globally explored points. In the initial portion of a PSO run, it performs more of exploration. However, as further generations evolve, the particles start converging, thus, making more of exploitation. Several strategies have been proposed in the literature [45], [46] to use multiple swarms within the overall process to strike a balance between exploration and exploitation. One such strategy, Locust swarm [45] is based on a "devour and move on" strategy – after a subswarm has found a local optima, a set of scouts are deployed to explore new promising regions. However, the scouts are guided by the intelligence gathered by the previous subswarms. We have utilized a similar strategy for better exploration of search-space.

For this we have run the *PSO* [34] several times to improve upon the global best solution. Suppose that the i^{th} run of the *PSO* produces the local best $pbest_i^k$ for each particle k and the global best $gbest_i$. In the $(i+1)^{th}$ run of the *PSO*, we start with a new set of particles. However, the global and local best information of the particles are passed from i^{th} to $(i+1)^{th}$ *PSO*. The number of times for which the *PSO* is run, that is, the terminating criteria is decided by the following.

(i) A user specified upper limit may be there. In our experimentation we have kept it at 200 individual *PSO* runs.

(ii) The global best cost does not improve in last 20 *PSO* runs.

2) *Initial population generations:* For initial population generation, we have used a modified Kernighan-Lin partitioning strategy [34], [47]. It has been originally developed to partition modules in VLSI physical design. It bipartitions a set of modules, so that highly connected modules are kept in one partition. In NoC, after the task to core mapping has been decided, an application consists of a set of cores carrying out specific tasks. The cores need to communicate between themselves, thus requiring bandwidths

for communication. Before going to the actual mapping, we propose to use a modified Kernighan-Lin (*KL*) bi-partitioning strategy to identify the closeness of cores by analyzing their bandwidth requirements. This bi-partitioning is applied (recursively) until the closest four cores are left in one final partition due to topological structure of BFT. The motivation for using this algorithm is that the cores with more communication requirement should be attached nearby routers in the NoC architecture. During the mapping phase, these partitions are taken into consideration to minimize the communication cost between mapped cores. The following algorithm is used to recursively partition the core graph.

Initially, at *level-0*, all cores are in one partition. At *level-1*, there are two partition sets, having partition numbers *0* and *1*, each containing half the nodes of the core graph. At next level (*level-2*), four partitions are generated (two each from partition-0 and partition-1 of level-1) having partition numbers *0, 1, 2* and *3*. This continues until there are only four cores ($\lambda = 4$) left in each partition for BFT. As KL partitioning results depend on the initial partitioning, we run the algorithm for L (preset) times, each time starting with a different randomly generated initial partition. Each initial partition gives different final partition having *4*-cores in each group. Those final partitions are used as particles for our *PSO*. The value of L is decided according to the number of particles we require for our *PSO*. In this process by using modified KL partitioning algorithm the initial population for our *PSO* is generated.

Since *KL* is a bi-partitioning strategy, we assume that the number of nodes in core graph is an exact power of *2*. Otherwise, we add a number of dummy nodes. Dummy nodes are connected to all core nodes and between themselves. Edges connecting a core node to dummy nodes are assigned cost zero while the edges between the dummy nodes are assigned cost infinity. Such a cost assignment favours the situation in which all dummy nodes will get clustered together. These dummy nodes are removed at the end of the initial mapping phase.

The algorithm takes a core graph $G = (C, E)$ as input, where C is the set of cores in the application and E is the set of edges depicting communication between them. It computes sets of clusters at different levels of partitioning. The *level-0* clusters, $clusters\ [0]$, is a single set consisting of all cores in C. They are partitioned into two sets at the next level. Thus $clusters\ [1]$ is a collection of two sets. The cores belonging to a set should be placed close to each other in the NoC, compared to two cores belonging to two different sets. In general, a set in $clusters\ [i]$ (that is, an i^{th} level cluster of cores) gives rise to two $i + 1$ level disjoint sets in $clusters\ [i + 1]$. The process continues till each set in a cluster contains only *2* cores for Mesh and MoT, and only *4* cores for BFT. The algorithm is invoked with *level=0*, *Cores_to_be_Partitioned = C*.

Algorithm KL-Partitioning

```
Input:   Core graph $G = (C, E)$: the core graph with set of
          cores $C$ and edges $E$
         $level$: level of partitioning
         $Cores\_to\_be\_Partitioned$: the subset of cores
          in $C$ to be partitioned
Output:   A set of clusters
          $clusters[i]$ is the $i^{th}$ level collection of
          disjoint sets of cores
Begin
      $clusters[level] = clusters[level] \cup Cores\_to\_be\_Partitioned$
      If $|Cores\_to\_be\_Partitioned| \le \lambda$ return     /* $\lambda = 2$
                                          for mesh and MoT, 4 for BFT*/
      $(p_1, p_2)$ = Random partitioning of cores in $Cores\_to\_be\_Partitioned$
      KL ($G, p_1, p_2$)
      KL-Partitioning ($G, level + 1, p_1$)
      KL-Partitioning ($G, level + 1, p_2$)
End
```

```
KL($G, p_1, p_2$)
Input: Core graph $G = (C, E)$: the core graph with set of cores
$C$ and edges $E$
$p_1, p_2$: non-empty equal-sized sets, such that $p_1 \cup p_2 = C$,
$p_1 \cap p_2 = \emptyset$
Output: updated $p_1, p_2$
Begin
     do
        $current\_partition = best\_partition = (p_1, p_2)$
        Unlock all cores
    While $unlocked\_cores\_exist(current\_partition)$ do
        swap = Select_next_move(current_partition)
        current_partition=Move_and_lock_cores(current_partition, swap)
        best_partition=Get_better_partition
                    (best_partition,current_partition)
    End While
        If not ($Cost\_fct\ (best\_partition) < Cost\_fct((p_1, p_2))$) then
          Return ($p_1, p_2$) // Terminate, no improvement
        Else // do another iteration
               ($p_1, p_2$) = $best\_partition$
               Unlock all cores
        End If
    End do
End
```

```
Select_next_move (P)
Begin
    For each unlocked ($c_i \in p_1, c_j \in p_2$) do
        Append ($costlog$, $CostFct$ ($Swap$ ($P, c_i, c_j$)))
    End For
        Return ($c_i, c_j$ swap in $costlog$ with lowest cost)
End
```

6. SIMULATION RESULT

In this section, we present the simulation results of our BFT application mapping techniques (Single *PSO* and Multiple augmented *PSO*), and compare with the solutions of other existing BFT mapping techniques for number of SoC benchmark applications. The core graphs for the applications are shown in Fig. 3.

6.1 Result on Communication Cost

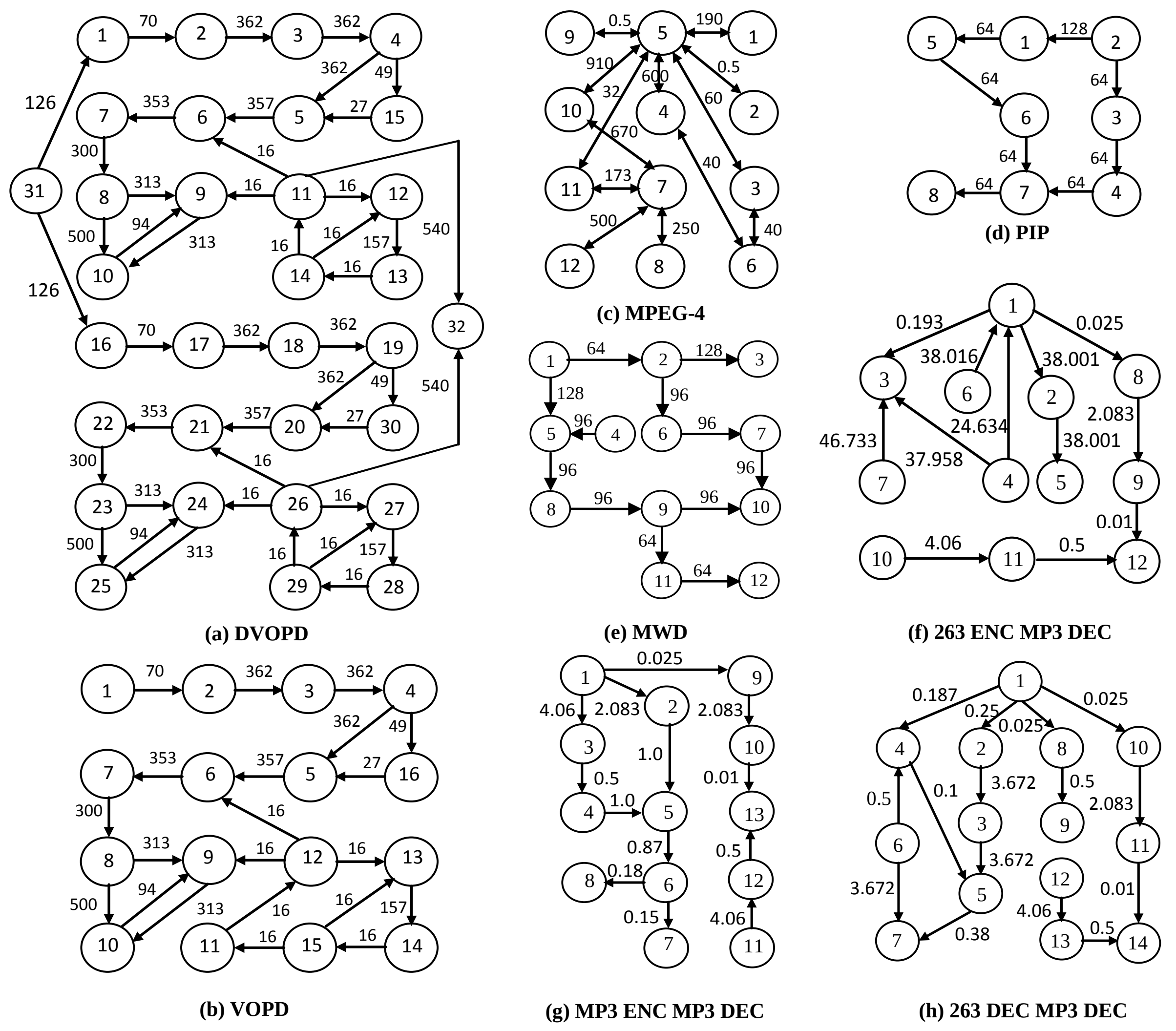
Communication cost is an indicative measure about the performance of a network. We have compared the *communication cost* for the benchmarks when mapped onto BFT network using our techniques and a mapping technique for BFT based NoC, KL_BFT [23].

Table 1 shows the comparative study across the mapping algorithms for the benchmarks. It can be noted that our BFT mapping technique with single *PSO* produces similar result for most of the benchmarks with reasonable CPU time. However, for DVOPD the result produced is inferior to KL_BFT [23]. The proposed BFT mapping technique with multiple augmented *PSO* produces the best result for each of the benchmarks with reasonable CPU time. This establishes the necessity of augmenting the basic *PSO* to achieve better results.

Table 1. Communication Cost and CPU Time for different Applications with their corresponding Techniques

Mapping Algorithms	DVOPD		VOPD		MPEG-4		PIP	
	Comm. Cost (bw × rc)	CPU time in s	Comm. Cost (bw× rc)	CPU time in s	Comm. Cost (bw × rc)	CPU time in s	Comm. Cost (bw × rc)	CPU time in s
KL_BFT [23]	24982	0.010	10498	~0.0	10144	~0.0	1536	~0.0
Our Single PSO	25162	0.650	10498	0.032	10144	0.012	1536	0.004
Our Multiple Augmented PSO	24806	2.090	10498	0.080	10144	0.060	1536	0.040

Mapping Algorithms	MWD		263enc mp3dec		mp3enc mp3dec		263dec mp3dec	
	Comm. Cost (bw × rc)	CPU time in s	Comm. Cost (bw× rc)	CPU time in s	Comm. Cost (bw × rc)	CPU time in s	Comm. Cost (bw × rc)	CPU time in s
KL_BFT [23]	3264	~0.0	561.876	~0.0	41.782	~0.0	42.180	~0.0
Our Single PSO	3264	0.020	561.876	0.024	41.782	0.060	42.180	0.016
Our Multiple Augmented PSO	3264	0.062	561.876	0.076	41.782	0.072	42.180	0.076



(a) DVOPD
(b) VOPD
(c) MPEG-4
(d) PIP
(e) MWD
(f) 263 ENC MP3 DEC
(g) MP3 ENC MP3 DEC
(h) 263 DEC MP3 DEC

Fig. 3. Application core graphs with communication bandwidth (MB/s) [12]

All the algorithms are run on an Intel *Core i5* platform with 4GB main memory and 2.4 GHz clock frequency. The CPU times needed in each of the techniques for individual benchmarks are noted in Table 1. The DPSO algorithm has been run with at most 200 particles for at most 100 generations without improvement.

6.2 Network Latency Comparison

For a better comparison among the mapping solutions, we have simulated each of the networks. The application traffic is generated by consulting the bandwidth requirements in different communications of the core graph. Synthetic self-similar traffic has been generated, guided by the communication requirements of cores in the application. Self-similar traffic has been observed in the burst traffic between on-chip modules in typical video and networking applications. A detailed description of such traffic generation strategy can be found in [48, 49]. We have used similar technique to generate traffic for our applications. A *SystemC* based simulator [10] has been utilized to compute the average network latency of the mapping solutions. $Latency$ refers to the length of time elapsed between the injection of a header flit at the source node and the reception of the tail flit at the destination. To reach destination node from source node, flits must travel through a path consisting of routers and interconnects. Depending on source/destination pair, each packet may encounter a different amount of $latency$. There are also some overheads at source and destination nodes which contribute to the overall latency. For packet i, the overall $latency$ can be defined as,

$$L_i = (Sender\ overhead) + (Transport\ latency) + (Receiver\ overhead)$$

So the $average\ latency$ will be,

$$L_{avg} = \frac{\sum_1^{TP} L_i}{TP}$$

where, TP be the total number of packets reaching their destination nodes. The *latency* is represented in terms of the number of *router cycles*. Table 2 shows the $latency$ results for each of the mappings by running the simulation for *2 00,000* clock cycles. Here also it can be observed that the augmented *PSO* produces least latency solutions.

Dynamic performance of the network in terms of latency, our proposed multiple augmented *PSO* technique produces better solutions than our single *PSO* technique and the BFT based NoC mapping technique proposed in KL_BFT [23] in almost all cases.

6.3 Energy Comparison

Energy consumed by the network is another determinant of the quality of mapping. Power/energy calculation depends on the power consumed by the routers and by the links. Energy consumption of links is determined separately from that of routers. In the absence of any information about sizes of individual cores, similar to [50], we have assumed each of them to be of dimension $2.5\ mm \times 2.5\ mm$. Copper wire (resistivity = *17 n*Ω*-m*) has been chosen as interconnection link. The width and thickness of the wire have been taken to be $0.25\ \mu m$ and $0.5\ \mu m$ respectively. The spacing between two adjacent wires is kept at $0.25\ \mu m$. The spacing between two adjacent metal layers is fixed at $0.75\ \mu m$ and is filled by a dielectric material having relative permittivity of 2.9 [10], [50]. Energy consumption for all possible transitions in the wires have been calculated using *HSPICE* [51].

Energy consumption of each router has been determined using *Synopsys Prime Power in 90 nm* CMOS technology with *Faraday* library by running their gate level netlists [52]. The number of toggles of every individual I/O pin of the router and their probability of remaining in *logic-1* state for the entire simulation window has been calculated from the NoC simulator. This information is then fed to *Synopsys Prime Power* tool [53] to estimate the power of each router with the following parameters: Process = *typical*, Supply Voltage = *1 V*, Temperature = *75 °C* and router clock period = *666 ps* [9–10], [50], [54]–[56].

Table 3 shows comparison of mapping solutions obtained via different approaches, in terms of energy consumption. Table 3, apart from reporting total network energy, also notes the average packet energy. As noted in [50], packet energy is an important attribute for characterizing NoC structures. With more packets traversing through the network, the total energy consumption will increase, however average packet energy is expected to reduce. This is what has precisely happened and is shown in Table 3.

From this table, it can be noted that our proposed multiple augmented *PSO* mapping technique results in less total network energy and average packet energy than single *PSO* technique and the BFT based NoC mapping technique proposed in KL_BFT [23] for all benchmark applications.

6.4 Bigger Application

To check the applicability of the *PSO* based approach on larger SoCs, we have used the *TGFF* tool [57] to generate a few task graphs with 64 and 128 cores. By varying bandwidth, number of start nodes and in-out degree for nodes, different task graphs have been generated via *TGFF*. The bandwidths are varied from $10\ MB/s$ to $1500\ MB/s$ for some graphs and $50\ MB/s$ to $150\ MB/s$ for other graphs. The in-out degrees of nodes are varied from 1 to 8 to generate both low and high communication graphs. Number of start nodes also varied to generate different graphs and to see the effect of mapping solutions upon them. Table 4 notes the mapping solutions for these task graphs. Here, it can be observed that our multiple augmented *PSO* based mapping technique produces better requiring, on an average, 0.91% and 12.37%, less communication than our single *PSO* mapping technique and KL_BFT [23] respectively within a reasonable CPU time.

7. CONCLUSION

In this paper we have presented mapping strategies for BFT based NoC using single *PSO* and multiple augmented discrete *PSO* technique. It can be noted from the results that, our multi-stage augmented *PSO* mapping strategy shows better result than our single-stage *PSO* mapping strategy and KL_BFT. For the NoCs having higher number of cores, our multi-stage augmented *PSO* mapping strategy produces the best solutions. Comparison of solutions produced establishes our mapping techniques onto BFT based NoC to be a strong competitor of previously available mapping strategies. The future scope of work includes trying out some actual system in which the mapping problem is solved using the multiple augmented *PSO* approach. The study of latency and energy profile of such a system can establish the effectiveness of the approach further.

Table 2. Average Network Latency in Router cycle

Mapping Algorithms	DVOPD	VOPD	MPEG-4	PIP	MWD	263enc mp3dec	mp3enc mp3dec	263dec mp3dec
KL_BFT [23]	104.85	103.60	90.80	83.50	90.30	92.50	84.30	86.50
Our Single *PSO*	104.80	103.58	90.75	83.50	90.25	92.40	84.22	86.40
Our Multiple Augmented *PSO*	104.55	103.50	90.60	83.50	90.10	92.20	84.10	86.30

Table 3. Communication Energy

Mapping Algorithms	Total Network Energy in μJ				Average Packet Energy in nJ			
	DVOPD	VOPD	MPEG-4	PIP	DVOPD	VOPD	MPEG-4	PIP
KL_BFT [23]	66.71	32.33	27.13	12.51	117.04	109.21	95.87	219.47
Our Single PSO	66.70	32.20	27.0	12.51	117.01	108.78	95.40	219.47
Our Multiple Augmented PSO	66.30	32.0	26.50	12.50	116.32	108.11	93.64	219.30

Table 4. Communication Cost and CPU Time for different TGFF Task Graphs with their corresponding

TGFF Task Graphs		KL_BFT [23]		Our single PSO		Our multiple augmented PSO	
		Comm. Cost (bw × rc)	CPU time in s	Comm. Cost (bw × rc)	CPU time in s	Comm. Cost (bw × rc)	CPU time in s
64 CORES	G1	20859.07	2.44	22990.39	6.97	20682.0	186.50
	G2	309416.59	0.70	324198.56	8.50	306714.0	185.0
	G3	299896.50	0.26	327510.21	7.30	298025.25	79.0
	G4	122950.40	2.50	135761.08	8.0	122708.65	222.0
	G5	15452.98	0.74	17303.88	10.66	15360.44	182.0
	G6	104187.66	1.42	113907.82	6.0	103317.12	42.0
128 CORES	G7	136413.67	4.32	158982.89	212.0	133257.65	821.40
	G8	1003356.75	3.97	1227588.37	233.37	991568.62	838.50
	G9	684005.81	3.37	839407.62	184.0	683725.19	435.0
	G10	173698.27	4.42	201500.26	272.0	170944.68	747.0
Average % of Improvement				over KL_BFT			0.91
				over Our single PSO			12.37

8. REFERENCES

[1] L. Benini, G. De Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70-78, 2002.

[2] W. J. Dally, B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proceedings of the Design Automation Conference (DAC)*, pp. 684-689, 2001.

[3] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, "A Network on Chip Architecture and Design Methodology," *Proceedings of ISVLSI*, pp. 117-124, 2002.

[4] S. Kundu, S. Chattopadhyay, "Interfacing Cores and Routers in Network-on-Chip Using GALS," *IEEE International Symposium on Integrated Circuits (ISIC)*, pp. 154-157, 2007.

[5] L. Benini, "Application Specific NoC Design," *IEEE Design, Automation and Test in Europe Conference (DATE'06),* vol. 1, pp. 1–5, 2006.

[6] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, F. Gebali, “A Topology-based Design Methodology for Networks-on-Chip Applications,” *In Proceedings of the second International Design and Test Workshop (IDT’07)*, pp. 61–65, 2007.

[7] P.P. Pande, C. Greca, M. Jones, A. Ivanov, R. Saleh, “Performance Evaluation and Design Trade-offs for MP-SOC Interconnect Architectures,” *IEEE Transactions on Computers*, vol. 54, no. 8, pp.1025–1040, 2005.

[8] A. O. Balkan, Q. Gang, U. Vishkin, “Mesh-of-Trees and Alternative Interconnection Network for Single-Chip Parallel Processing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, pp. 1419-1432, 2009.

[9] K. Manna, S. Chattopadhaya, I. Sengupta, “An Efficient Routing Technique for Mesh-of-Tree-based NoC and its Performance Comparison,” *International Journal of High Performance Systems Architecture*, vol. 4, no. 1, pp 25-37, 2012.

[10] S. Kundu, S. Chattopadhyay, “Design and Evaluation of Mesh-of-Tree based Network-on-Chip using Virtual Channel Router,” *Journal of Microprocessors and Microsystems, Elsevier*, vol. 36, issue 6, pp. 471–488, 2012.

[11] N. Koziris, M. Romesis, P. Tsanakas, G. Papakonstantinou, “An Efficient Algorithm for the Physical Mapping of Clustered Task Graphs onto Multiprocessor Architectures,” *Proceedings of 8th EuroPDP*, pp. 406-413, 2000.

[12] P. K. Sahu, S. Chattopadhyay, “A Survey on Application Mapping Strategies for Network-on-Chip Design,” *Journal of Systems Architecture, Elsevier*, vol. 59, 2013, pp. 60-76.

[13] J. Hu, R. Marculescu,“Energy-Aware Mapping for Tile-based NOC Architectures Under Performance Constraints,” *ASP-DAC*, pp. 233-239, 2003.

[14] S. Murali, G. De Micheli, “Bandwidth Constrained Mapping of Cores onto NoC Architectures,” *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE)*, vol. 2, pp. 896-901, 2004.

[15] K. Srinivasan, K. S. Chatha, A Technique for Low Energy Mapping and Routing in Network-on-Chip Architecture, *IEEE International Symposiun on Low Power Electronics and Design (ISLPED)*, pp. 387-392, 2005.

[16] T. Shen, C. H. Chao, Y. K. Lien, and A. Y. Wu, “A New Binomial Mapping and Optimization Algorithm for Reduced-Complexity Mesh- based on-Chip Network,” *Proceedings of NOCS’07*, pp. 317-322, 2007.

[17] R. Mehran, S. Saeidi, A. Khademzadeh, A. A. Kusha, “Spiral: A Heuristic Mapping Algorithm for Network on Chip,” *IEICE Electronics Express*, vol. 4, no. 15, pp. 478-484, 2007.

[18] M. Janidarmian, A. Khademzadeh, M. Tavanpour, “Onyx: A New Heuristic Bandwidth-Constrained Mapping of Cores onto Network on Chip,” *IEICE Electronics Express*, vol. 6, no. 1, pp. 1-7, 2009.

[19] M. Tavanpour , A. Khademzadeh, M. Janidarmian, “Chain-Mapping for Mesh based Network-on-Chip Architecture,” *IEICE Electronics Express*, vol. 6, no. 22, pp. 1535-1541, 2009.

[20] Y. Chen, L. Xie, J. Li, “An Energy-Aware Heuristic Constructive Mapping Algorithm for Network on Chip,” *International Conference on ASIC (ASICON)*, pp. 101-104, 2009.

[21] M. Reshadi, A. Khademzadeh, A. Reza, “Elixir: A New Bandwidth-Constrained Mapping for Networks-on-Chip,” *IEICE Electronics Express*, vol. 7, no. 2, pp. 73-79, 2010.

[22] S. Tosun, “New Heuristic Algorithm for Energy Aware Application Mapping and Routing on Mesh-based NoCs,” *Journal of System Architecture, Elsevier*, 57, pp. 69-78, 2011.

[23] P. K. Sahu, K. Manna, T. Shah, and S. Chattopadyay, “Extending Kernighan–Lin partitioning heuristic for application mapping onto Network-on-Chip,” *Journal of System Architecture, Elsevier*, vol. 60, pp. 562-578, 2014.

[24] P. K. Sahu, N. Shah, K. Manna, S. Chattopadhyay, “A New Application Mapping Algorithm for Mesh based Network-on-Chip Design,” *IEEE International Conference (INDICON)*, pp. 1-4, 2010.

[25] P. K. Sahu, K. Manna, T. Shah, and S. Chattopadyay, “Thermal Uniformity-Aware Application Mapping for Network-on-Chip Design,” *International Journal of Computer Applications*, vol. 99 (2), pp. 8-22, 2014.

[26] T. Lei, S. Kumar, “A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture,” *Proceedings of the Euromicro Symposium on Digital System Design (DSD)*, pp. 180-187, 2003.

[27] K. Bhardwaj, R. K. Jena, “Energy and Bandwidth Aware Mapping of IPs onto Regular NoC Architectures Using Multi-objective Genetic Algorithms,” *International Symposium on System-on-Chip (SOC)*, pp. 27-31,2009.

[28] F. M. Darbari, A. Khademzadeh, G. G. Fard, “CGMAP: A New Approach to Network-on-Chip Mapping Problem,” *IEICE Electronics Express*, vol. 6, no. 1, pp. 27-34, 2009.

[29] G. Fen, W. Ning, “Genetic Algorithm based Mapping and Routing Approach for Network on Chip Architectures,” *Chinese Journal of Electronics*, vol. 19, no. 1, pp. 91-96, 2010.

[30] M. Tavanpour, A. Khademzadeh, S. Pourkiani, M. Yaghobi, “GBMAP: An Evolutionary Approach to Mapping Cores onto a Mesh-based NoC Architecture,” *Journal of Communication and Computer*, vol. 7, no. 3, pp. 1-7, 2010.

[31] W. Zhou, Y. Zhang, Z. Mao, “Link-load Balance Aware Mapping and Routing for NoC, WSEAS Transactions on Circuits and Systems,” vol. 6, issue 11. pp. 583-591, 2007.

[32] W. Lei, L. Xiang, “Energy- and Latency-Aware NoC mapping Based on Discrete Particle Swarm Optimization,” *Proceedings of IEEE Internationa Conference on Communications and Mobile Computing*, pp. 263-268, 2010.

[33] A. H. Benyamina, P. Boulet, A. Aroul, S. eltar, K. Dellal, “Mapping Real Time Applications on NoC Architecture

with Hybrid Multi-objective Algorithm," *International Conference on Metaheuristics and Nature Inspired Computing*, pp. 1-10, 2010.

[34] P. K. Sahu, P. Venkatesh, S. Gollapalli, S. Chattopadhyay, "Application Mapping onto Mesh Structured Network-on-Chip using Particle Swarm Optimization," *IEEE International symposium on VLSI (ISVLSI)*, pp. 335-336, 2011.

[35] P. K. Sahu, T. Shah, K. Manna, and S. Chattopadhyay, "Application Mapping onto Mesh based Network-on-Chip using Discrete Particle Swarm Optimization," *IEEE Transactions on VLSI Systems (T-VLSI)*, vol. 2, issue 22, pp. 300-312, 2014.

[36] J. Wang, Y. Li, S. Chai, Q. Peng, "Bandwidth-Aware Application Mapping for NoC-Based MPSoCs," *Journal of Computational Information Systems*, 7:1, pp. 152-159, 2011.

[37] P. K. Sahu, N. Shah, K. Manna, S. Chattopadhyay, "An Application Mapping Technique for Butterfly-Fat-Tree Network-on-Chip," *IEEE International Conference on Emerging Applications and Information Technology (EAIT)*, pp. 383-386, 2011.

[38] P. K. Sahu, N. Shah, K. Manna, S. Chattopadhyay, "A New Application Mapping Strategy for Mesh-of-Tree based Network-on-Chip," *IEEE International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT)*, pp. 518-523, 2011.

[39] P. K. Sahu, A. Sharma, S. Chattopadhyay, "Application Mapping onto Mesh-of-Tree based Network-on-Chip using Discrete Particle Swarm Optimization," *IEEE International Symposium on Electronic System Design (ISED)*, pp. 172-176, 2012.

[40] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, "High-Throughput Switch-based Interconnect for future SoCs," *IEEE International Workshop on System-on-Chip for Real Time Applications*, pp. 304–310, 2003.

[41] I. Kennedy, R. C.Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks*, NJ. pp.1942-1948, 1995.

[42] K. Wang, L. Huang, C. Zhou, W. Pang, "Particle Swarm Optimization for Traveling Salesman Problem," *Proceedings of the Second International Conference on Machine Learning and Cybermetics*, pp. 1583-1585, 2003.

[43] Yuhui Shi, Russell Eberhart, "Parameter Selection in Particle Swarm Optimization," *Springer Berlin/ Heidelberg*, vol. 1447/1998, pp. 591-600, 2006.

[44] L. Guilan, Z. Hai, S. Chunhe, "Convergence Analysis of a Dynamic Discrete PSO Algorithm," *International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pp. 89-92, 2008.

[45] A. B. Röhler, S. Chen, "An Analysis of Sub-swarms in Multi-swarm Systems," *Proceedings of Joint Australasian Conference in Artificial Intelligence*, Springer-Verlag, pp. 271–280, 2011.

[46] S. Chen, J. Montgomery "Selection Strategies for Initial Positions and Initial Velocities in Multi–optima Particle Swarms," *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 53–60,2011.

[47] B. Kernighan, S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[48] G. V. Varatkar, R. Marculescu, "On-Chip Traffic Modelling and Synthesis for MPEG-2 Video applications," *IEEE Trasactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, issue. 1, pp. 108-119, 2004.

[49] K. C. Chang, T. F. Chen, Low-power Algorithm for Automatic Topology Generation for Application-specific Networks on Chips, *IET Computers & Digital Techniques*, vol. 2, no. 3, pp. 239-249, 2008.

[50] B. S. Feero, P. P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32-45, 2009.

[51] "HSPICE Reference Guide", Version U-2003.09, September 2003, Synopsys Inc.

[52] "Design Vision User Guide", Version U-2003.03, March 2003, Synopsys Inc.

[53] "Synopsys Prime Power Mannual", Version Y-2006.06, June 2006, Synopsys Inc.

[54] C. A. Nicopoulor, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, C. R. Das, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," *IEEE/ACM Symposium on Micro architecture*, pp. 333-346, 2006.

[55] S. Pasricha, N. Dutta, "On-chip Communication Architectures: System on Chip Interconnect," *Morgan Kaufmann Publishers*, Chapter: 5, Page: 176, Table 5.7: 2008.

[56] S. Traboulsi, N. Pohl, J. Hausner, A. Bilgic, V. Frascola, "Power Analysis and Optimization of the ZUC Stream Cipher for LTE-Advanced Mobile Terminals," *IEEE Latin American Symposium on Circuits and Systems*, pp. 1-4, 2012.

[57] R. P. Dick, D. L. Rhodes, W. Wolf, "TGFF: Task Graphs For Free," *Proceedings of International Workshop on Hardware/Software Codesign*, 1998.