Thermal Uniformity-Aware Application Mapping for Network-on-Chip Design

Pradip Kumar Sahu[1], Kanchan Manna[2], Tapan Shah[3] and Santanu Chattopadhyay[4]
Electronics and Electrical Communication Engineering,
Indian Institute of Technology, Kharagpur, West Bengal, India

ABSTRACT

Ensuring thermal-uniformity in an integrated circuit chip is very essential for its correct operation. Thus, in the Network-on-Chip (NoC) based system design as well, it is essential to attach cores of the application core graph to the routers in the topology graph so that thermal uniformity across the chip is maintained. However, the performance of the application should not be sacrificed to a great extent. Also, the CPU time needed to explore the overall search-space is quite high. This paper presents a tool to the designers to explore the search-space in a controlled fashion. The designer can specify the communication cost degradation that can be tolerated and the amount of effort put in to identify the potential solutions. All non-dominated solutions (in terms of communication cost and temperature variance) are reported from which the designer can choose the appropriate one for implementation.

General Terms

Thermal-aware application mapping, Heuristic, Network-on-Chip

Keywords

Application mapping, Communication cost, Mesh topology, Network-on-Chip, Temperature variance, Thermal uniformity

1. INTRODUCTION

In recent years, power density in processor has doubled every three years. This rate is expected to increase further within next one or two generations, due to the higher rate of shrinking feature size, increasing transistor count, and faster frequency scaling, compared to the reduction in operating voltage [1]. High-performance circuits consume large amount of power due to their increased bandwidth requirement, higher frequency of operation, and higher level of system integration. A good amount of consumed power is converted directly into dissipated heat. Such a system must be designed to ensure good thermal behavior of the chip, even when the maximum power is dissipated by it. A major concern in today's system design is the thermal heating of ICs. As Network-on-Chip (NoC) consists of different cores, each having its own power-profile, area, frequency of operation etc, it results in non-uniform heating of the chip. This may result in delay variation across the chip. This not only affects circuit performance but also decreases their reliability. Hence, ensuring thermal uniformity across the chip is a necessity. Excessive localized heating occurs much faster than chip-wide heating. Since power dissipation is non-uniform across the chip, this leads to the creation of thermal hotspots that can cause timing errors or even physical damage. One solution to this problem is the usage heat sink and some other cooling techniques. As a result, various cooling solutions have been proposed in the literature. As power consumption increases, there is a non-linear relationship between the cooling capabilities and the cost of the solution [2]. Apart from heat sink and cooling strategies, another solution to the problem is the placement of cores – the placement should be guided not only by their communication requirements, but also their temperature profile.

A major challenge in thermal uniformity-aware NoC based system design is to determine the association of routers of the fabric to the cores of an application. An application consists of a set of tasks, each of which is implemented by an IP core. As the tasks need to interchange messages between themselves, so do the IP cores. After the cores participating in an application have been decided, the application can be represented in the form of a *core graph* [3], 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.

On the other hand, the given NoC topology can be represented in the form of a *topology graph* [3].

Definition 2: The NoC *topology graph* 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, $map: C \to U, such\ that, \forall c_i \in C, \exists u_j \in U\ and\ map(c_i) = u_j$.

The function associates core c_i to router u_j. Naturally, assuming that at most one core can be attached to each router, 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. The communication between each pair of cores can be treated as flow of a single commodity $d^k, k = 1,2,\ldots,|E|$. The value of commodity d^k, corresponding to the communication between cores c_i and c_j is equal to $comm_{i,j}$, the bandwidth requirement. If c_i is mapped to the router $map(c_i)$ and c_j is mapped to $map(c_j)$, the set of all commodities $D = \{d^k\}$ is defined as follows.

$$D = \{d^k | value(d^k) = comm_{i,j}, for\ k = 1,2,\ldots,|E| \text{ and } e_{i,j} \in E\} \quad (1)$$

Also,

$$source(d^k) = map(c_i) \text{ and } sink(d^k) = map(c_j) \quad (2)$$

The link between two individual routers u_i and u_j of the topology has a maximum bandwidth of $bw_{i,j}$. The total commodity flowing through such a link should not exceed this bandwidth. The quantity $x_{i,j}^{k}$ indicating the value of commodity d^k flowing through the link (u_i, u_j) is given by,

$$x_{i,j}^{k} = \begin{cases} value\ (d^k), if\ link\ (u_i, u_j) \in Path\ \left(source\ (d^k), sink\ (d^k)\right) \\ 0, otherwise \end{cases} \quad (3)$$

where, $Path\ (a, b)$ indicates the deterministic routing path between the router nodes a and b in the topology. Satisfaction of bandwidth limitations of individual links must be ensured. That is, all mapping solutions should satisfy the following relation.

$$\sum_{k=1}^{|E|} x_{i,j}^{k} \leq bw_{i,j}\ , for\ all\ i, j \in \{1, 2, \ldots, |U|\} \quad (4)$$

If all bandwidth constraints are satisfied, the $communication\ cost\ T_x$ of a mapping solution is given by,

$$T_x = \sum_{k=1}^{|E|} value\ (d^k) \cdot hopcount\ (source\ (d^k), sink\ (d^k)) \quad (5)$$

Here, $hopcount\ (a, b)$ is the number of hops between the topology nodes a and b. For a deterministic shortest path routing, $hopcount$ corresponds to the minimum number of hops between the constituent nodes. Since $communication\ cost$ is very much dependent on the mapping solution, the overall mapping problem is often formulated to optimize the $communication\ cost$, ensuring that the bandwidth constraints of all individual links are satisfied. $Communication\ cost$ affects the performance of the overall system and its energy consumption, as both of these factors are directly proportional to the total $hopcount$.

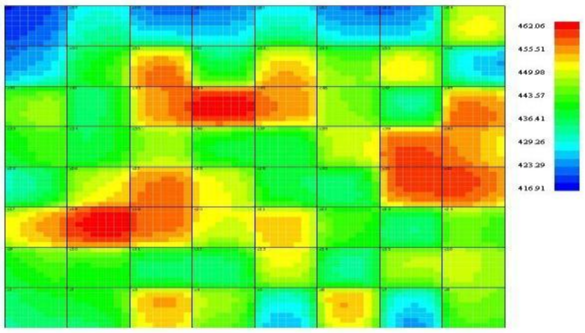

Fig.1. Thermal profile for TGFF graph G3 (Communication-aware mapping [7])

Communication cost = 107888.02
Temperature variance = 146.83
Peak Temperature = 462.06°*K*

Several application mapping algorithms have been proposed in the literature to minimize the communication cost and energy consumption of NoC [4]. However, algorithms which minimize communication cost of the mapping may not consider the thermal effects, resulting in hotspots and high peak temperatures. It may also create very high temperature variance within the chip, resulting in uneven delay across the chip. This paper presents the design of our proposed mapping algorithm to minimize both communication cost and temperature variance for a given application. Temperature variance and peak temperature should be reduced with a limitation on permissible communication cost trade-off. Our mapping strategy has been developed for mesh topology, though it can very easily be extended to any other topologies. In the following, this scenario has been discussed with an example application core graph (named G3) generated using *TGFF* [5], the random task graph generator. The *TGFF* tool [5] have been used the to generate a few task graphs with 64 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 $10MB/s$ to $1500MB/s$ for some graphs and $50MB/s$ to $150MB/s$ 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. The bandwidth values for the edges are also generated randomly to get heterogeneous communication behavior of cores. *64*-core NoCs are implemented as *8×8*.

A thermal simulation using HotSpot tool [6] has been applied upon a communication-aware application mapping of the core graph (G3) using discrete Particle Swarm Optimization [7]. Power densities of the cores are generated randomly within 10–60 (W/cm^2) [8]. Router power values are calculated using their switching activities. Temperature profile of the communication-aware mapping of the application graph G3 has been shown in Fig. 1. The resulting communication and temperature metrics are also noted in the figure.

Here temperature variance is the summation of squared differences of individual tile temperatures from the average chip temperature.

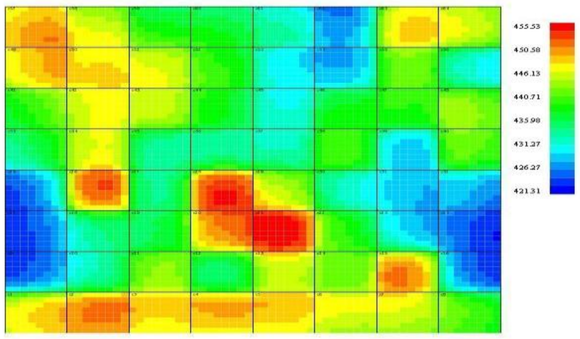

Fig.2. Thermal profile for TGFF graph G3 (Thermal uniformity-aware mapping)

Communication cost = 119782.0
Temperature variance = 110.78
Peak Temperature = 455.53°*K*

$$T_{var} = \sum_{i=1}^{N} (T_{tile_i} - T_{avg})^2 \quad (6)$$

Where, T_{var}= Temperature variance of chip

T_{tile_i}= Temperature of i^{th} tile inside the chip
T_{avg}= Average temperature of chip
N = Total number of tiles inside the chip

Peak temperature T_{peak} is the maximum temperature of a tile inside the chip.

$$T_{peak} = Max(T_{tile_i}),\ \text{for}\ i = 1, 2, \ldots, N \quad (7)$$

As shown in Fig.1, there is quite a large variation in temperature across the chip, almost *45°*. Such a wide variation may create a non-uniform delay at different regions of the chip. This motivates us to go for a thermal uniformity-aware application mapping. The goal of the strategy is to reduce the temperature variance across the chip. Peak temperature of the chip also gets reduced in some cases. The temperature profile of thermal uniformity-aware mapping of G3 application benchmark obtained using the technique discussed in this paper is shown in Fig. 2. Temperature variance is reduced by almost *25%*, but the communication cost increases by *11%.* As shown in Fig. 2, the peak temperature is also reduced by *1.5%.*

This paper presents a thermal uniformity-aware application mapping strategy onto mesh-based NoC using a constructive heuristic to make the temperature profile uniform, as well as reduce the peak temperature with tolerable performance degradation. The salient features of the approach are as follows.

1. A trade-off has been established between performance and temperature variance of the system.
2. The technique is flexible in the sense that the user can set the tolerance limit for the communication cost in their design. The amount of effort the mapper employs (thus, the CPU time) is also controllable.

The rest of the paper is organized as follows. Section 2 surveys the works reported in the literature on NoC mapping techniques. Section 3 discusses the relevant issues in mapping algorithm design. Section 4 presents our proposed thermal uniformity-aware mapping algorithm. Section 5 embodies the results and analysis of the proposed approach. Conclusion is presented in Section 6.

2. LITERATURE SURVEY

To obtain optimum solution to the application mapping problems, several researchers have proposed Integer Linear Programming (ILP) based formulations [9–14]. While [9] attempts to minimize energy by shutting down certain communication links in NoC based chip multiprocessors (CMPs), a unified approach of energy efficient application mapping has been presented in [10] taking care of all the sub-problems, such as, application mapping, operating voltage assignment, and routing. In [11], the existing ILP [10] has been extended to find a trade-off between computation and communication energy. In [12], factors that produce network contention have been analyzed. It proposes an ILP formulation for a contention-aware application mapping algorithm in tile-based NoC to minimize inter-tile network contention. In [13, 14], authors have presented ILP formulation for application mapping onto mesh based NoC to minimize energy consumption for different benchmarks.

PMAP, a two-phase mapping algorithm for placing clusters onto processors has been presented in [15], where highly communicating clusters are placed on adjacent nodes of the processor network. In [3], 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. In [16, 17], 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. MOCA, a two phase heuristic for low energy mesh based on-chip interconnection architecture has been proposed in [18]. A binomial IP mapping and optimization algorithm (BMAP) has been presented in [19] to reduce hardware cost of the on-chip network. Spiral, a mapping algorithm has been proposed in [20] which reduce the cumulative energy consumption of communication links and the overall system execution time. In [21], adaptive feedback control based NoC architecture with multiple voltage clocks for multiple islands has been proposed to minimize the power consumption by exploiting dynamic voltage-frequency scaling. Onyx, a bandwidth constrained application mapping has been presented in [22] to minimize the overall communication cost of NoC. CHMAP [23] 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 [24] is a constructive application mapping algorithm that maps cores onto NoC minimizing total communication cost and energy. In [25], authors have taken NMAP [3] as their initial mapping solution. A branch-and-bound algorithm, as in [16], 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 [26].

A two-step Genetic Algorithm (*GA*) for mapping applications onto NoC has been proposed in [27], which reduces the overall execution time. A multi-objective Genetic Algorithm (MOGA) based application mapping technique has been proposed in [28], 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 [29], 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 [30], 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 [31], which reduces energy consumption and total bandwidth requirement of NoC. PLBMR, a Particle Swarm Optimization (*PSO*) based two-phase application mapping algorithm proposed in [32] minimizes the NoC communication energy and allocates the routing path for balancing the link-load. A mapping technique based on discrete *PSO* has been presented in [33]. However, it only considers improvement over genetic algorithm based method and reports relative improvements only. In [34], 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 [35], a meta-heuristic strategy using *PSO* technique has been proposed to reduce both static and dynamic cost of NoC for mesh based application mapping. A discrete multiple *PSO* based mapping technique has been proposed [7] 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.

The above mapping techniques do not consider the temperature effect during mapping. Temperature affects performance, power, and reliability of the system. A temperature-aware task mapping and scheduling technique has been proposed in [37], which maps tasks using a heuristic and a floorplanning tool to reduce the peak temperature. Power densities are expected to increase faster in future technologies, as the operating voltage no longer scales as quickly as it has. The International Technology Roadmap for Semiconductor (ITRS) Document (2003) [1] has projected very little change in operating voltage. A good amount of work has been proposed to design new packages that provide good heat removal capacity and arrange circuit boards to improve air flow. Chips are packaged with die placed against a spreader plate, often made of aluminium, copper or some other highly conductive material, which is in turn placed against a heat sink of aluminium or copper that is cooled by a fan [38]. On the thermal modeling of ICs, HotSpot [6] is an accurate and automated fast thermal estimation tool that calculates transient temperature response, given the physical characteristics and power of units on the die. It is based on an equivalent circuit of thermal resistances and capacitances that correspond to micro-architecture blocks and essential aspects of the thermal package.

Dynamic thermal management (DTM) for MPSoC or NoC refers to hardware and software strategies which work dynamically, at run-time, to control different IP cores operating temperature. On the other hand, in case of static thermal management (STM), generally the thermal management is performed off-line, before the application is run and at the time of application mapping. The DTM techniques for MPSoC or NoC have been proposed to reduce the thermal packaging and cooling costs. It controls overheating by keeping the temperature below a critical threshold. Computation migration and fetch toggling are examples of such techniques [38]. To overcome the limitations of worst case thermal management, a distributed dynamic thermal management scheme, called ThermalHerd has been proposed in [39], [40] for on-chip networks, which can dynamically regulate the network temperature profile and guarantee safe on-line operations with little performance impact. Many thermal management techniques have been proposed to reduce the overall power consumption of the chip. However, there are some localized temperature problems in NoC, referred to as hotspots. In [41], authors have proposed a hotspot prevention technique that dynamically reconfigures the functionalities at runtime across the IPs in order to balance the temperature profile. The temperature management with software techniques, especially OS-level task scheduling has been proposed in different literatures for both single and multi-core processors. A thermal aware dynamic OS-level work-load scheduling have been proposed in [42] to get better thermal profile with negligible performance overhead. In this technique, when temperature reaches the critical value, a heuristic is applied to distribute the workload for better temporal and spatial temperature distribution.

In case of static thermal management, the thermal balancing is done before the application is run, that is, at the time of core placement, depending on communication requirement among cores of MPSoC or NoC and their temperature profile. The physical location of cores, the load of each core, and the communication across the cores of a NoC, contribute to the power consumption and are directly related to hotspots [43]. An IP virtualization and placement technique based on Genetic Algorithm (GA) has been proposed in [44] for regular NoC architectures, which attempts to achieve a thermal balance while minimizing the communication cost via placement. IP virtualization, which maps the logic processing unit onto processing elements (PEs), thus allowing the PE to virtually perform the computation and communication, affects the power consumption and the communication cost. The mapping problem formulated in [44] is a three-objective optimization problem – communication cost, energy consumption, and thermal balance. Communication cost and energy consumption are related to $hopcount$ of NoC. A pareto based mapping technique using Genetic Algorithm has been proposed in [45], which minimizes the average $hopcount$ and achieves thermal balance. In NoC, power is dissipated when packets traverse through switches and links. The dominating power consuming operations are buffer reads and writes, switching, routing decisions, channel allocation and link utilization. A systematic methodology, such as, application independent power-aware routing algorithm and buffer sizing for NoC, targeting temperature reduction has been proposed in [46]. This achieves significant peak temperature reduction. The cores having more communication volume should be mapped close to each other for minimization of communication cost and energy consumption. Because of high communication volume these cores have high temperature and easily cause creation of hotspot regions. A multi-objective ant colony algorithm (MOACA) has been proposed in [47] that maps IP cores onto mesh based NoC, which optimizes energy consumption and thermal balance.

In some thermal management techniques, thermal balancing is done before the application is run as well as it includes some hardware and software strategies which work dynamically, at run-time, to control the operating temperature of different IPs. A temperature-aware thermal management technique has been proposed in [48] for thermal balancing of MPSoC. In this technique, authors have first performed an integer linear programming (ILP) based static task scheduling for minimization of energy consumption and reduction of hotspots. Then an OS-level dynamic scheduling as in [42] has been applied upon it to arrive at a better thermally balanced solution. A temperature-aware task mapping algorithm has been proposed in [49] to prevent hotspot in MPSoC platform. Next, uniform thermal distribution has been performed using adaptive multi-threshold values during run-time. In this technique, the algorithm keeps track of the temperature of the cores, and swaps the mapped tasks when the core temperature is relatively higher than average chip temperature. The cores may be switched off if they exceed an absolute maximum temperature. For power reduction in NoCs, different voltage-frequency selection techniques have been proposed.

The strategies proposed above come up with a single solution. Ideally, the synthesis process should be able to explore a large number of alternatives and report non-dominated solutions, when the solutions are judged from performance and thermal angles. This paper attempts to bridge the gap by generating a good number of solutions depending upon the amount of tolerable performance sacrifice and computational effort that the designer is ready to pay. The most suitable solution from this set can be chosen by the designer.

3. RELEVANT ISSUES IN MAPPING ALGORITHM DESIGN

A thermal uniformity-aware mapping strategy has to take care of both the resulting temperature distribution and the overall communication cost. A strategy attempting communication overhead reduction will try to put highly communicating cores close to each other, while thermal-aware strategies will attempt to put cores consuming high power, far away. The amount of emphasis to be put on either aspect will have its effect on the solution quality. In the following a discussion has been presented on the same, from the three different angles.

A. *Core sequencing* – the order in which the cores should be picked up for mapping by the algorithm.
B. *Objective function formulation* – the relative weight to be put on the two aspects of optimization.
C. *Communication cost tolerance* – that allows the algorithm to explore solutions with more thermal uniformity around one with almost similar communication cost.

Table 1: Communication cost and temperature variance of different core sequencing strategies

TGFF Graphs	Strategy-1			Strategy-2			Strategy-3			Constructive Mapping		
	Comm. cost	Temp. variance	Peak temp.	Comm. cost	Temp. variance	Peak temp.	Comm. cost	Temp. variance	Peak temp.	Comm. cost	Temp. variance	Peak temp.
G1(64)	10623.10	126.50	454.14	13280.0	128.31	457.37	14744.40	157.84	460.15	6734.78	184.58	463.07
G2(64)	137389.0	153.47	458.0	158329.0	111.65	453.56	192636.0	165.92	457.95	113653.0	185.38	462.43
G3(64)	143073.0	151.05	458.04	152999.0	112.34	455.33	186612.0	153.51	457.92	109327.0	156.01	458.76
G4(64)	56544.40	138.94	457.80	60123.40	172.77	462.92	64984.90	160.72	462.63	48765.40	174.17	463.79
G5(64)	7912.20	134.93	457.78	8600.08	120.14	459.53	8567.50	171.78	464.25	5933.51	175.55	464.30
G6(64)	55068.30	145.65	460.68	61470.10	115.29	455.81	66713.20	146.28	461.04	42086.60	146.98	461.08
G7(64)	9639.14	173.82	436.72	12553.30	101.54	451.94	14134.80	174.26	464.02	6259.47	179.78	465.55

A. Core Sequencing

Any heuristic mapping procedure essentially works with an ordering of cores (either implicitly or explicitly) to pick up the successive cores for mapping to the routers. Thus, a major challenge in the development of thermal-aware mapping strategy is to identify a technique to order the cores, so that the dual purpose of ensuring thermal uniformity and minimizing communication cost could be achieved. To evolve such a strategy, it's first tried with the following three alternatives that put full emphasis on the power consumed by the cores, disregarding the communication requirements.

Strategy 1: Sort cores in descending order of power consumption and map consecutive cores at close proximities.

Strategy 2: Sort cores in descending order of their power consumption and try to map consecutive cores away from each other.

Strategy 3: Alternately select one high power consuming core and a low power consuming one. Map consecutive cores in this order close to each other.

To get the mapping solutions picking up cores in either of these strategies, a constructive algorithm has employed, similar to the initial population generation policy in [7]. Only the core ordering is depicted by the strategy. The results produced show very good improvement in thermal variation over a purely communication cost aware strategy. The communication-aware strategy uses the same algorithm, however, the cores are picked up in decreasing communication requirement order. The communication overhead increases significantly, in three strategies making all of them unacceptable (Table 1). However, this shows that the choice of core order affects the final solution significantly and warrants the development of a good ordering strategy. In the following, the strategy followed in this work has been presented.

To identify a good sequence it proceeds as follows. First, it computes the sequence, p_{seq} of cores sorted in descending order of their power consumption values. Communication cost of such a solution is quite high, compared to a communication-aware mapping (as demonstrated in Table 1). For an application with N cores, it next generates N sequences, each starting with a unique core. The sequence $N_{seq}[core\_num]$ is the sequence of cores in which the core, $core\_num$ appears as the first one. In such a sequence, the next core selected is the highest communicating one with the core $core\_num$. In general, if already mapped i^{th} cores from the sequence, the $(i+1)^{th}$ core is selected to be the highest communicating one with these i^{th} cores. The procedure is illustrated in Procedure *Find_Core_Sequence*.

For each of N sequences $N_{seq}[i]$, it computes distance $dist_i$, defined as follows.

$$dist_i = \sum_{k=1}^{N} \left| index\left(N_{seq}[i], k\right) - index\left(p_{seq}, k\right) \right| \qquad (8)$$

where $index(s, k)$ is the index of core k in sequence s. Thus, $dist_i$ is a measure of similarity between the sequences $N_{seq}[i]$ (sequence with core i as the starting core) and p_{seq} (decreasing order of power sequence). If the distance is high, sequence $N_{seq}[i]$ is almost the opposite of the p_{seq} sequence. If the distance is low, $N_{seq}[i]$ almost resembles the p_{seq} sequence. Both these sequences are unacceptable. Hence, it chooses the sequence which is equal or nearest to the average distance. Such a sequence will resemble a mix of high and low power consuming cores. This is expected to be beneficial for both thermal and communication cost minimization. The process has been described in procedure *Find_Best_Sequence*.

```
Procedure Find_Core_Sequence
Input:   Core graph G,
Output:  Core sequences $N_{seq}$
Begin
        While all cores are not selected as staring core do
            Select core core_num
            Count = 0
            $N_{seq}$ [core_num][Count] =core_num
            While there exists unselected cores in G do
                Count ++
                Let ($C_i$, $C_j$) be the edge width highest
                communication requirement
                such that exactly one of $c_i$ and $c_j$ is
                already selected
                Set $c= c_i$ if Cj is already selected Else Set $c=c_j$
                $N_{seq}$ [core_num]=c
                Mark c selected
            End While
        End While
End
```

```
Procedure Find_Best_Sequence
Input: Core sequences $N_{seq}$, Core sequence according to
decreasing power consumption $P_{seq}$
Output: The best sequence
Begin
        $Avg_{dist}= 0$
        For each sequence $N_{seq}[i]$ do
        $dist_i = \sum_{k=1}^{N} |index(N_{seq}[i], k) - index(p_{seq}, k)|$
        $Avg_{dist} = Avg_{dist} + \frac{dist_i}{N}$
        End For
        $Min_{dist} = \infty$
        $i = 0$
        For each sequence $N_{seq}[i]$ do
                If $(|Avg_{dist} - dist_i| < Min_{dist})$ then
                        $Start\_Core = i$
                        $Min_{dist} = |Avg_{dist} - dist_i|$
                End if
                $i++$
        End For
        Return $N_{seq}[Start\_Core]$
End
```

B. Objective Function Formulation

The algorithm gives weight to the temperature variance and communication cost of a solution, in order to explore the search space by using the following cost function.

$$val = wt * \left(\frac{T_{var}}{BT_{var}}\right) + (1 - wt) * \left(\frac{Comm\_Cost}{\text{Best\_cost}}\right), 0 \le wt \le 1 \qquad (9)$$

where, T_{var} = Temperature variance of the mapping
BT_{var} = Temperature variance when *wt = 1*
Comm_Cost = Communication cost of mapping
Best_cost = Communication cost of mapping when *wt = 0*

The value *wt=1* puts full emphasis on the temperature variance minimization, while *wt=0* emphasizes communication cost reduction. Our algorithm produces a set of solutions corresponding to different weights. The designer has the option to select any one from the reported solutions. More the number of *wt* values explored, higher will be the execution time of the algorithm. In order to limit it, user needs to input the parameter *Effort* (the number of different *wt* values to be explored). The *wt* values are updated as,

$$wt = wt + \frac{1}{Effort+1} \qquad (10)$$

The value *Effort = 0* explores two values of *wt* (*0* and *1*).

C. Communication Cost Tolerance

One problem with the cost function in eq. (9) is that it performs a trade-off between improvement in temperature variance and degradation in communication cost. Since the designer may not be willing to have solutions with high communication costs (even though the thermal behavior is very good), it is desirable to have another degree of control over communication cost degradation. The parameter *tolerance_limit* is the allowed percentage degradation in communication cost that the user is ready to sacrifice with respect to a fully communication-aware mapping (that is, *wt = 0*). To take this into consideration, the algorithm first performs a pure communication-aware mapping. In subsequent runs, the algorithm considers candidate solutions with communication costs within the tolerance limit of this mapping only.

4. MAPPING ALGORITHM

The algorithm starts with *wt = 0*, which optimizes only the communication cost. Next, mapping is done with *wt = 1*, which optimizes only the temperature variance having communication cost degradation within the tolerance limit. Now, two solutions can be – one with reduced communication cost and the other with reduced temperature variance. Next, it find solutions intermediate to these values. First it selects the core sequence using the procedure *Find_Best_sequence*. The number of solutions produced by the algorithm is dependent on the *Effort* value provided by the user. If *Effort* is less, algorithm iterates for less number of times exploring small number of solutions. If *Effort* is more, the algorithm will iterate for more number of times, exploring more solutions. Each time the algorithm iterates, it uses a different value of *wt.* As *Effort* increases, algorithm tries to find good solutions using different *wt* values. Finally it outputs all *non-dominated* solutions. A *non-dominated* solution is one which is better than all others in either the communication cost or the temperature variance. The designer can select any of these solutions for mapping.

Algorithm *Map_Graph* calls the function *Do_Mapping_Cal_Cost* to obtain all non-dominated mapping solutions generated by putting the first core in a given sequence onto each of the router positions. The solutions are produced via the function *Find_Mapping*. Once all such solutions have been generated, *Do_Mapping_Cal_Cost* reports only the non-dominated ones for the perusal of the designer.

The procedure *Find_Mapping* finds mapping of core graph *G* onto topology graph *P*. This procedure takes starting core and starting router position to start mapping. It maps the first core to the starting router position provided. It selects next core from the N_{seq}*[start_core]* sequence and finds positions nearby to the already mapped cores. It constructs the set of positions consisting of all router positions, one-hop away from any of the mapped cores. The communication costs for each of these positions is evaluated (using function *Evaluate_Positions*) and the router positions giving minimum communication cost are copied into the set *Min_Postions*. If the set *Min_Positions* contains a single entry, the corresponding router position is taken to be the *Best_Position* for the core. Otherwise, the procedure *Predict_Best* is called to get the best position. The procedure *Predict_Best* attempts to distinguish between the router positions in *Min_Positions* set. To evaluate the fitness of one such router position in *Min_Positions*, it places the core under consideration at that position and then continues to place the remaining cores in *G* with the core under consideration placed at the candidate router position. Fitness is computed as a weighted sum of temperature variance and communication cost. However, only these solutions qualify for fitness calculation for which the degradation in communication cost is within the tolerance limit from *Best_cost* (communication cost corresponding to a totally communication-aware mapping). The function returns the best position which is suitable for optimizing either the communication cost, or the temperature variance, or both. Mapping continues in similar fashion with remaining cores. Function *Compute_comm_cost* computes the communication cost of the mapping produced using eq. (5) (Section I). The procedure *Find_Thermal_Cost* computes the temperature profile of the NoC using the power trace and the chip layout. The tool HotSpot [6] has been used to generate the temperature map. Temperature variance and peak are calculated from the generated temperature information.

Algorithm: **Map_Graph**

```
Input:   Core graph $G$, Topology graph $P$, Power consumed by
each core $Core\_power$
Output: Mapping of $G$ onto $P$
enum $optimization$ {$Cost\_Opt, Temp\_Opt, Both\_Opt$} $flag$;
Begin
Sort edges of $G$ descending order of communication cost
$Set\ wt = 0$
Let $(c_1, c_2)$ be the edge of $G$ with the highest required bandwidth
$Cost_1 = \sum_{c_i \in neighbour(c_1)}$ Bandwidth requirement of $(c_1, c_i)$
$Cost_2 = \sum_{c_i \in neighbour(c_2)}$ Bandwidth requirement of $(c_2, c_i)$
If $(Cost_1 > Cost_2)$ then $Start\_Core = c_1$ Else $Start\_Core = c_2$
$BT_{var} = 1; Best\_cost = 1; flag = Cost\_Opt$
Do_Mapping_Cal_Cost $(G, P, Start\_Core, flag, wt)$
Find_Core_Sequences $(G)$
$Start\_Core =$ Find_Best_Sequence()
$flag = Temp\_Opt$
$wt = 1$
Do_Mapping_Cal_Cost $(G, P, Start\_Core, flag, wt)$
$flag = Both\_Opt$
$wt = 0$
    For $i = 1$ to $Effort$ do
        $wt = wt + \frac{1}{Effort + 1}$
        Do_Mapping_Cal_Cost $(G, P, Start\_Core, flag, wt)$
    End For
End
```

Procedure: Do_Mapping_Cal_Cost

```
Input: Core graph $G$, Topology Grapg $P$
        $Start\_Core$: with which core mapping should start
        $flag$: which optimization should be achieved
        $wt$: weight for optimization of objective function
Output: Minimum value of fitness $Min\_val$, Best mapping
solution $Best\_mapping$ enum $optimization\ flag$
Begin
    For each router position $u$ of $P$ do
        Mark all cores of $G$ as unmapped
        $Min\_val = \infty$
        $Best\_mapping = \emptyset$
        $Mapping$ =Find_mapping ($G,P,u,$Start_core)
        If ($flag == Cost\_Opt$ || $flag == Both\_Opt$)
            $Cost$=Compute_Comm_cost($Mapping, G$)
        If($flag == Temp\_Opt$ || $flag == Both\_Opt$)
            $T_{var}$=Find_Thermal_Cost($Mapping, Core\_Power$);
    End For
    Output all non-dominated mapping solutions
End
```

Procedure: **Find_Mapping**

```
Input: Core graph $G$, Topology graph $P$
        $Start\_Posn$: Position in $P$ where first core to be mapped
        $Start\_Core$: With which core mapping should start
Output: Mapping of all cores of $G$ onto $P$ with the $Start_{Core}$
        mapped to $Start\_Posn$
Begin
    $Mapping\ [Start\_Posn] = Start\_Core$
    Mark $Start\_Core$ as mapped
    While there exist unmapped cores in $G$ do
        $c$=Next core from the $N_{seq}[Start\_Core]$
        Position=set of position in P with one hop
                distance from already mapped
                positions
        Evaluate_Positions ($Positions$)
        Min_Positions = Set of Positions with minimum
                communication cost
        If (cardinality of set $Min\_Positions == 1$)
            $Best\_Position = Min\_Positions\ [0]$
        Else
            $Best\_Position$ = Predict_best($G,P,Min\_$
                $Positions, Start\_Core$)
        End If
        Mapping[Best_position] = c
        Mark $c$ mapped
    End while
    Return $Mapping$
End
```

Procedure Predict_Best

```
Input:  Core graph $G$, Topology graph $P$, Core to be mapped $c$,
        contending router positions $Min_{Positions}$core sequence
        identified by $Start\_Core$
Output: Predicted minimum communication cost position for $c$
Begin
    Set $Temp\_marked = \emptyset$ /* Holds the cores marked
                temporarily*/
    For each position $p \in Min\_Positions$ do
        $Mapping\ [c] = p$ /* Map $c$ to $p$*/
        Mark $c$ as mapped
        $Temp\_marked = Temp\_marked \cup \{c\}$

    While there exists unmapped cores in $G$ do
        $c =$ Next unmapped core
            from the $N_{seq}[Start\_Core]$
        $Positions$ = Router positions in $P$ one hop
                away from mapped positions
        Evaluate_Positions ($Positions$)
        $Best_{Position} =$ First Position with
                minimum communication cost
        $Mapping\ [Best\_Position] = c$
        Mark $c$ mapped
        $Temp\_marked = Temp\_marked \cup \{c\}$
    End while
    $Comm_{Cost} =$ Total communication
            cost for this mapping
        If (wt == 0) /* Communication-aware mapping*/
            Set fitness of $P = Comm\_Cost$
        Else /* $wt \neq 0$*/
          If $Comm_{Cost} > \left(1 + \frac{tolerance_{limit}}{100}\right) \times Best\_cost$
                                        /* Discard the
                                        solution*/
            Set fitness of $P = \infty$
          Else /* Consider both communication and
            temperature factors*/
            $T_{var} =$ Find_Thermal_Cost ($Mapping$)
            $val = wt * \left(\frac{T_{var}}{BT_{var}}\right) + (1 - wt) * \left(\frac{Comm\_Cost}{Best\_cost}\right)$
            Set fitness of $P = val$
          End If
        End If
        Unmark all cores in $Temp\_marked$
    End For
    Return router position $p$ corresponding to
    minimum $fitness$ value
End
```

Procedure Find_Thermal_Cost

```
Input: mapping of core to NoC $Mapping$
       Core power information $core\_power$
Output: Temperature variance of layout $T_{var}$
Begin
       Create power trace file()
       Generate thermal profile()   /* Using the tool
                                    HotSpot 5.01*/
       $T_{var} =$ Get temperature variance()
       Return $T_{var}$
End
```

5. SIMULATION RESULTS

HotSpot 5.01 [6] has been used to generate the temperature profile after mapping the cores onto the routers of a NoC. HotSpot requires some basic input files, such as, floor plan of the given architecture and the power consumption profile of each block to generate the temperature profile. Hotspot generates temperature profile using these values and parameters specified in a configuration file. The important parameter values for configuration are mentioned in Table 2. The file generated by HotSpot tool is inspected for temperature variance and peak temperature of the chip. The model used for simulation in HotSpot is the block model. Power consumption of the cores are generated randomly between 10-60 (*W/cm*^{2}) [8] and router power values are calculated by their switching activities. Temperature variance is calculated using eq. (6), and the peak temperature using eq. (7).

This algorithm has been tested on several graphs and results have been noted. First, it has calculated results for different tolerance and *Effort* values for application core graphs generated by *TGFF* [5]. Table 3 summarizes the results.

Fig. 3 shows the solutions generated by different *Efforts* as shown in Table 3 for different tolerance values. From Fig. 3, it can be noted that in case of *5%* and *10%* tolerances, there is a significant difference over temperature variance but limited number of solutions are generated due to less tolerance allowed for communication cost. As shown in Fig. 4, in case of *20%* tolerance, the solution points are parabolic in nature,

Table 2. HotSpot parameters

Chip Specifications	
Chip thickness in meter	0.00015
Silicon thermal conductivity in W/(m-K)	100.0
Silicon specific heat in J/(m^3-K)	1.75×10^6
Temperature threshold for DTM (Kelvin)	354.95
Heat Sink Specifications	
Convection capacitance in J/K	140.4
Convection resistance in K/W	0.1
Heat sink side in meter	0.06
Heat sink thickness in meter	0.0069
Heat sink thermal conductivity in W/(m-K)	400.0
Heat sink specific heat in J/(m^3-K)	3.55×10^6
Heat Spreader Specifications	
Spreader side in meter	0.03
Spreader thickness in meter	0.001
Heat spreader thermal conductivity in W/(m-K)	400.0
Heat spreader specific heat in J/(m^3-K)	3.55×10^6
Interface Material Specifications	
Interface material thickness in meter	2.0×10^5
Interface material thermal conductivity in W/(m-K)	4.0
Interface material specific heat in J/(m^3-K)	4.0×10^6

which is required for such an optimization. There is a good difference in temperature variance as well as communication cost. For *30%* tolerance the solution quality does not improve over *20%.* This happens due to creation of large number of intermediary solutions which appear to be promising at the time of their generation but do not yield good complete solutions. Fig. 5 shows the CPU time for different tolerance values with *Efforts 7*. It can be noted that CPU time increases with increase in tolerance values. From *20%* to *30%*, there is a stiff increase in CPU time. However, from *5%* to *20%* tolerance there is almost a linear increment of the CPU time requirement. Therefore, from Fig. 4 and Fig. 5, choosing *20%* tolerance appears to be a good trade-off between solution quality and CPU time.

Hence, it has chosen *20%* tolerance value for all subsequent results. *Effort* also plays important role in finding good solutions. As *Effort* value increases, algorithm finds more number of good solutions. Table 4 shows the communication cost, corresponding temperature variance and peak temperature for different TGFF application graphs with different *Effort* values for *20%* tolerance.

From Table 4, it can be noted that for *Effort 7,* healthy solutions have been found. Fig. 6 shows solutions for different *Effort* values. For less *Effort*, very few solutions are generated and also the solution quality is poor. The observation is more prominent in Fig. 7. However running the algorithm for higher *Effort* is time consuming. Fig. 8 shows the CPU time behavior for different efforts. From this, it is clear that as *Effort* increases, CPU time also increases almost linearly.

Table 5 notes comparison among communication cost optimization (*wt=0*), temperature optimization (*wt=1*) and both communication cost and temperature optimization (*wt=0.5*) of different TGFF graphs after mapping onto NoC.

Table 3. Different tolerance values and *Effort* values for TGFF graph G3 (64 cores)

Effort	Tolerances of Communication Cost in %			
	5	10	20	30
	(Comm. Cost, Temp. Variance)	(Comm. Cost, Temp. Variance)	(Comm. Cost, Temp. Variance)	(Comm. Cost, Temp. Variance)
1	(113895, 147.09) (114577, 140.39)	(114269, 127.01)	(111572.0, 134.44) (114247.0, 118.33) (119782.0, 110.78) (123274.0, 108.35)	(116063, 120.58) (116536, 116.54) (116983, 115.56) (117029, 114.43) (126949, 112.70) (130985, 107.35)
3	(112058, 154.06) (113895, 147.09) (114577, 140.39)	(112715, 132.02) (114269, 127.01)	(111572.0, 134.44) (112572.0, 124.54) (114247.0, 118.33) (118768.0, 111.64) (119782.0, 110.78) (122655.0, 108.96) (123274.0, 108.65)	(110948, 130.20) (126949, 112.70) (130060, 112.44)
5	(112058, 154.06) (113895, 147.09) (114577, 140.39)	(112715, 132.02) (114269, 127.01) (115379, 126.74)	(111387.0, 136.74) (111572.0, 134.44) (112572.0, 124.54) (112769.0, 123.10) (114247.0, 118.33) (118768.0, 111.64) (119782.0, 110.78) (122655.0, 108.96) (123274.0, 108.65)	(110078, 137.13) (126949, 112.70) (130060, 112.44)
7	(112058, 154.06) (113895, 147.09) (114577, 140.39)	(111599, 136.15) (114269, 127.01) (115825, 120.24) (119093, 117.08)	(111387.0, 136.74) (111572.0, 134.44) (112572.0, 124.54) (112769.0, 123.10) (114247.0, 118.33) (118768.0, 111.64) (119782.0, 110.78) (122655.0, 108.96) (123274.0, 108.65)	(110078, 137.13) (120576, 114.35) (126949, 112.70) (130060, 112.44)

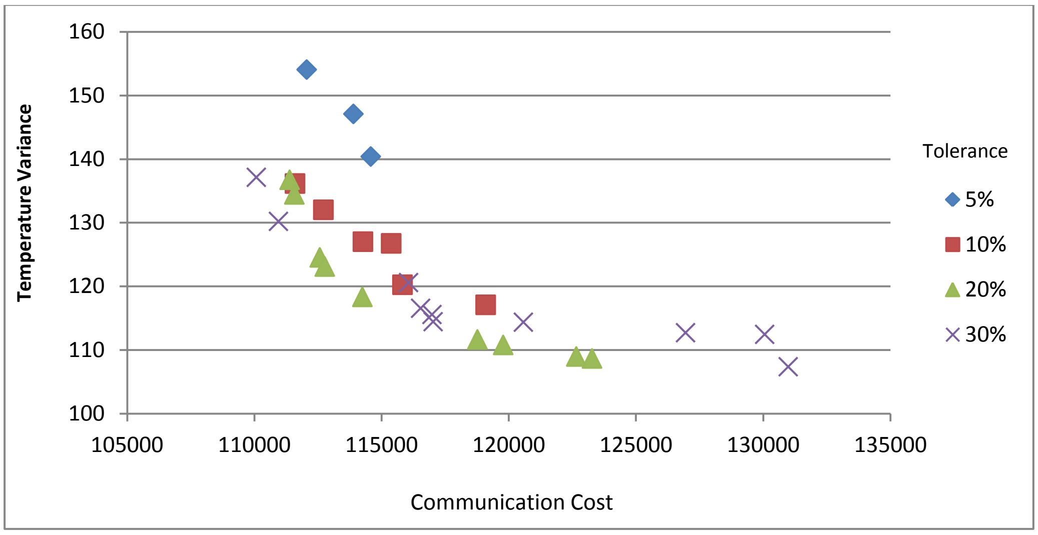

Fig 3: Communication cost vs. temperature variance of different tolerance values for G3

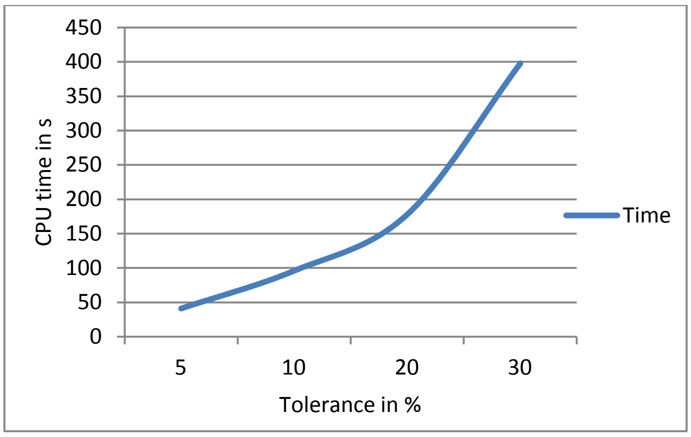

Fig 4: CPU time at different tolerance values for G3

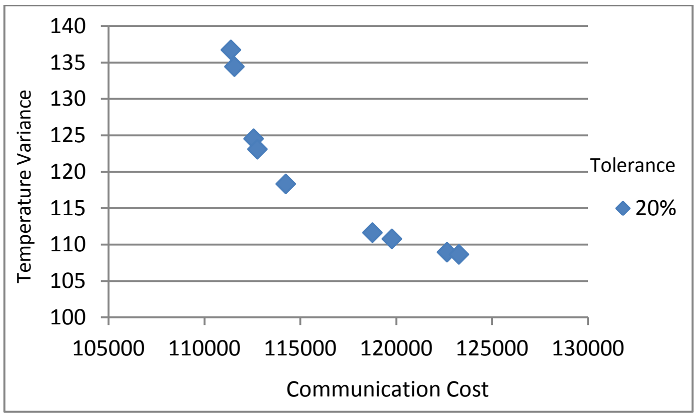

Fig 5: Communication cost vs. temperature variance for 20% tolerance for G3

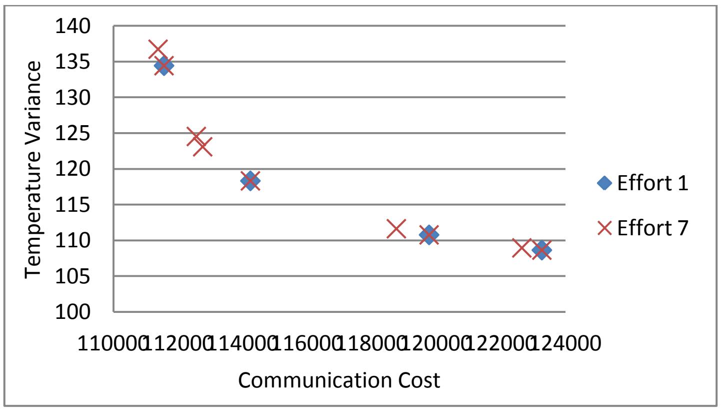

Fig 6: Solutions for different *Effort* values of TGFF graph G3

From the table it can be noted that improvement in temperature variance is quite high, up to *30%* for most of the cases and degradation in communication cost is limited to *20%* as specified by the user.

Table 6 notes the communication cost and temperature variance of different *TGFF* graphs after mapping onto NoC using different mapping techniques. Here also it can be noted that our proposed thermal uniformity-aware algorithm using temperature optimization technique and combined temperature and communication cost optimization technique produce better temperature variance than other mapping techniques. Last row of Table 6 shows the communication cost of different mapping techniques normalized with respect to [7]. It also shows temperature variances normalized with respect to the thermal uniformity-aware mapping with weight *wt = 1.0*. The same information has also been plotted in Fig. 9 and 10 respectively. The thermal mapping produces good saving in temperature variance compared to the other mapping methodologies.

6. CONCLUSION

This paper proposed a thermal uniformity-aware application mapping strategy to reduce temperature variance as well as peak temperature with tolerable communication cost degradation. In this mapping technique a trade-off has been established between performance and temperature variance of the system. The technique is flexible in the sense that the user can set the tolerance limit for the communication cost degradation in their design. The parameter *Effort* can also be varied to generate more number of solutions. Thus, it provides a performance- and thermal-aware application mapping tool for NoC design. As this problem is NP-complete, evolutionary based optimization technique can be used for better result.

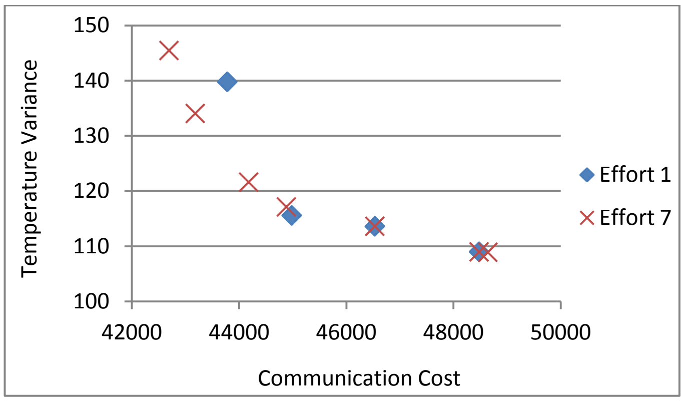

Fig 7: Solutions for different *Effort* values of TGFF graph G6

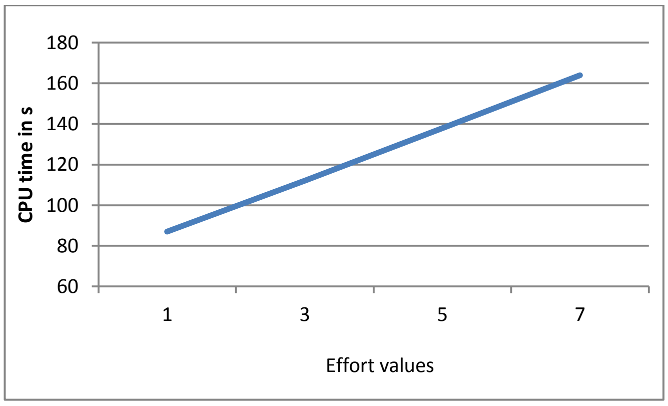

Fig 8: CPU time behavior for different *Effort* values

Table 4. Results for different TGFF graphs and different *Effort* values for 20% tolerance

TGFF Application Graphs	*Effort 1*	*Effort 3*	*Effort 5*	*Effort 7*
	(Comm. Cost, Temp. Variance,Peak Temp.)	(Comm. Cost, Temp. Variance, Peak Temp.)	(Comm. Cost, Temp. Variance, Peak Temp.)	(Comm. Cost,Temp. Variance, Peak Temp.)
G1 (64 cores)	(7284.27, 134.29, 460.38) (7886.17, 125.81, 457.57)	(6988.64, 170.13, 462.83) (7284.27, 134.29, 460.38) (7886.17, 125.81, 457.57)	(6988.64, 170.13, 462.83) (7284.27, 134.29, 460.38) (7886.17, 125.81, 457.57)	(6988.64, 170.13, 462.83) (7284.27, 134.29, 460.38) (7326.99, 133.70, 459.77) (7886.17, 125.81, 457.57)
G2 (64 cores)	(133467.0, 160.73, 459.53) (134613.0, 152.16, 459.26)	(132452.0, 174.83, 459.84) (133467.0, 160.73, 459.53) (134613.0, 152.16, 459.26)	(132452.0, 174.83, 459.84) (133467.0, 160.73, 459.53) (133753.0, 155.15, 459.37)	(132452.0, 174.83, 459.84) (133085.0, 165.06, 459.62) (134613.0, 152.16, 459.26)
G3 (64 cores)	(111572.0, 134.44, 458.39) (114247.0, 118.33, 456.09) (119782.0, 110.78, 455.53) (123274.0, 108.65, 454.15)	(111572.0, 134.44, 458.39) (112572.0, 124.54, 458.19) (114247.0, 118.33, 456.09) (118768.0, 111.64, 456.06) (119782.0, 110.78, 455.53) (122655.0, 108.96, 454.31) (123274.0, 108.65, 454.15)	(111387.0, 136.74, 458.47) (111572.0, 134.44, 458.39) (112572.0, 124.54, 458.19) (112769.0, 123.10, 457.68) (114247.0, 118.33, 456.09) (118768.0, 111.64, 456.06) (119782.0, 110.78, 455.53) (122655.0, 108.96, 454.31) (123274.0, 108.65, 454.15)	(111387.0, 136.74, 458.47) (111572.0, 134.44, 458.39) (112572.0, 124.54, 458.19) (112769.0, 123.10, 457.68) (114247.0, 118.33, 456.09) (118768.0, 111.64, 456.06) (119782.0, 110.78, 455.53) (122655.0, 108.96, 454.31) (123274.0, 108.65, 454.15)
G4 (64 cores)	(54634.80, 173.02, 462.39) (54680.80, 150.33, 462.30) (55644.10, 141.60, 461.43) (57647.20, 122.10, 459.43)	(54634.80, 173.02, 462.39) (54680.80, 150.33, 462.30) (54829.80, 144.73, 462.24) (55007.30, 141.70, 462.19) (56206.30, 141.12, 461.09) (57647.20, 122.10, 459.43)	(53830.80, 175.55, 462.58) (54680.80, 150.33, 462.30) (54829.80, 144.73, 462.24) (55007.30, 141.70, 462.19) (56206.30, 141.12, 461.09) (57647.20, 122.10, 459.43)	(53152.60, 176.11, 463.05) (54680.80, 150.33, 462.30) (54829.80, 144.73, 462.24) (55007.30, 141.70, 462.19) (56206.30, 141.12, 461.09) (57647.20, 122.10, 459.43)
G5 (64 cores)	(6833.53, 143.81, 461.82) (6836.40, 142.01, 461.59) (6939.91, 140.55, 460.34) (7101.82, 139.06, 460.26)	(6811.29, 162.49, 461.92) (6826.01, 144.21, 461.91) (6939.91, 140.55, 460.34) (7101.82, 139.06, 460.26)	(6811.29, 162.49, 461.92) (6826.01, 144.21, 461.91) (6939.91, 140.55, 460.34) (7101.82, 139.06, 460.26)	(6787.21, 172.09, 462.61) (6811.29, 162.49, 461.92) (6826.01, 144.21, 461.91) (6939.91, 140.55, 460.34) (7101.82, 139.06, 460.26)
G6 (64 cores)	(43782.10, 132.77, 459.31) (44984.50, 115.57, 454.25) (46534.80, 113.60, 455.03) (48480.10, 108.98, 453.23)	(42695.30, 145.45, 460.33) (43782.10, 132.77, 459.31) (44180.0, 123.61, 458.84) (44885.60, 117.11, 456.01) (48480.10, 108.98, 453.23)	(42695.30, 145.45, 460.33) (43184.30, 140.04, 460.14) (43245.10, 139.13, 460.06) (44791.0, 121.87, 456.65) (44885.60, 117.11, 456.01) (48480.10, 108.98, 453.23)	(42695.30, 145.45, 460.33) (43184.30, 140.04, 460.14) (44180.0, 121.61, 458.84) (44885.60, 117.11, 456.01) (46534.80, 113.60, 455.03) (48480.10, 108.98, 453.23)
G7 (64 cores)	(7153.55, 120.92, 457.30) (7203.07, 118.88, 456.77) (7288.39, 118.20, 456.71)	(6868.89, 177.09, 465.42) (6982.50, 133.17, 459.82) (7153.55, 120.92, 457.30) (7203.07, 118.88, 456.77)	(6868.89, 177.09, 465.42) (6982.50, 133.17, 459.82) (7045.54, 130.32, 459.10) (7153.55, 120.92, 457.30) (7203.07, 118.88, 456.77)	(6868.89, 177.09, 465.42) (6982.50, 133.17, 459.82) (7108.58, 127.96, 458.04) (7153.55, 120.92, 457.30) (7203.07, 118.88, 456.77) (7288.39, 118.20, 456.71)

Table 5. Optimization results for different graphs with 20% tolerance

TGFF Graphs	Comm. Cost Optimization (*wt = 0*)		Temp. Optimization (*wt = 1.0*)		Comm. Cost and Temp. Optimization (*wt = 0.5*)		% of change for *wt=1.0* w.r.t. *wt=0*		% of change for *wt=0.5* w.r.t. *wt=0*	
	Comm. Cost	Temp. Variance	Comm. Cost	Temp. Variance	Comm. Cost	Temp. Variance	Comm. Cost Degradation	Temp. Variance Improvement	Comm. Cost Degradation	Temp. Variance Improvement
G1	6734.78	184.58	7886.17	125.81	7886.17	125.81	17.10%	31.84%	17.10%	31.84%
G2	113653.0	185.38	133753.0	155.15	133085.0	165.06	17.69%	16.31%	17.10%	10.96%
G3	109327.0	156.01	123274.0	108.65	119782.0	110.78	12.76%	30.36%	9.56%	28.92%
G4	48765.40	174.17	57647.20	122.10	57647.20	122.10	18.21%	29.90%	18.21%	29.90%
G5	5933.51	175.55	7101.82	139.06	6939.91	140.55	19.69%	20.79%	16.96%	19.94%
G6	42086.60	146.98	48480.10	108.98	44885.60	117.11	15.19%	25.85%	6.65%	20.32%
G7	6259.47	179.78	7153.55	120.92	7153.55	120.92	14.28%	32.74%	14.28%	32.74%

Table 6. Communication cost and temperature variance of different mapping techniques

TGFF Graphs	NMAP [3]		LMAP [50, 51]		PSO based Mapping [7]		Comm. Cost and Temp. Optimization (*wt = 0.5*)		Temp. Optimization (*wt = 1.0*)	
	Comm. cost	Temp. variance	Comm. cost	Temp. variance	Comm. cost	Temp. variance	Comm. cost	Temp. variance	Comm. Cost	Temp. Variance
G1	9207.50	149.66	7441.40	165.40	6734.78	184.12	7886.17	125.81	7886.17	125.81
G2	132292.38	185.73	120209.59	159.43	107741.99	185.52	133085.0	165.06	133753.0	155.15
G3	116337.81	143.69	118879.44	193.06	107888.02	146.83	119782.0	110.78	123274.0	108.65
G4	55244.17	123.79	49694.11	155.94	45598.64	159.95	57647.20	122.10	57647.20	122.10
G5	6015.28	161.05	6267.59	174.32	5758.61	145.69	6939.91	140.55	7101.82	139.06
G6	44902.16	171.46	42394.97	165.09	40905.62	154.89	44885.60	117.11	48480.10	108.98
G7	9129.94	166.32	7655.83	201.48	6210.72	162.09	7153.55	120.92	7153.55	120.92
Normalized Average	1.214	1.261	1.110	1.404	1.0	1.304	1.176	1.024	1.198	1.0

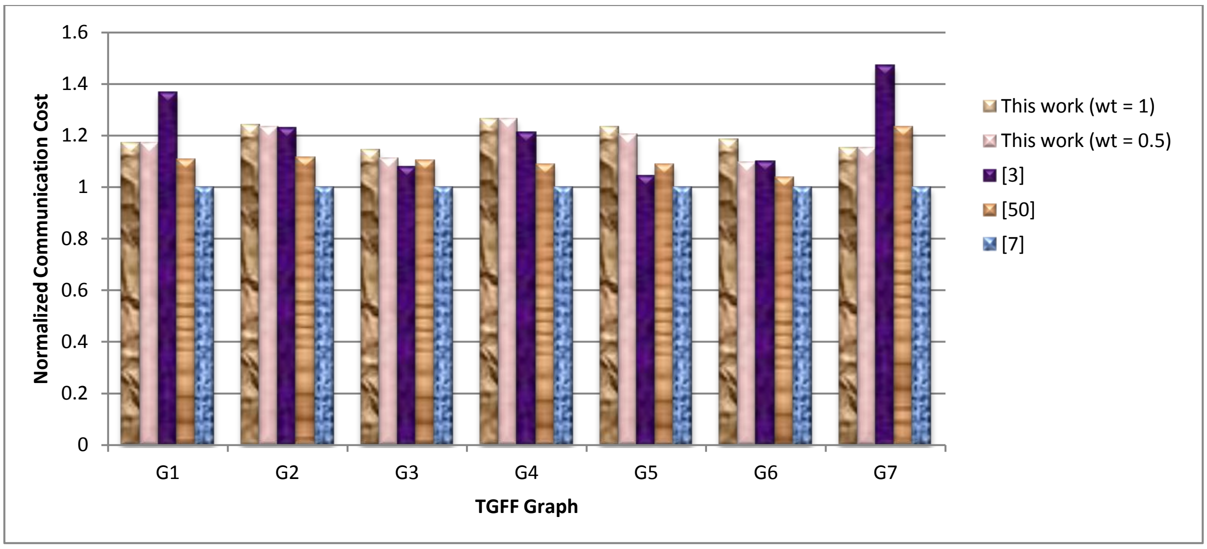

Fig 9: Normalized communication cost of different mapping techniques

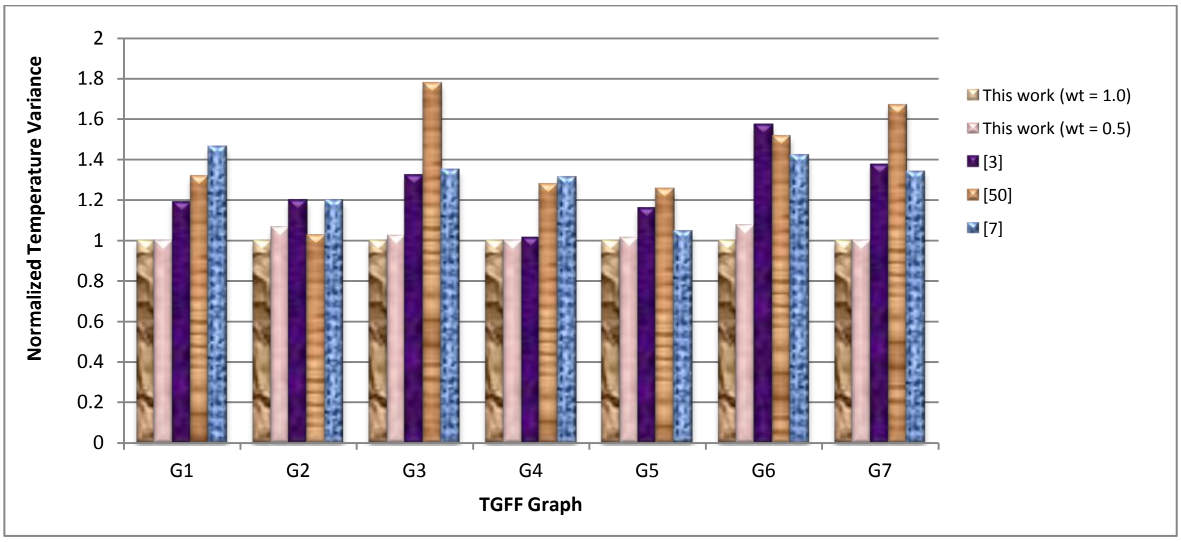

Fig 10: Normalized temperature variance of different mapping techniques

7. REFERENCES

[1] 2003 International Technology Roadmap for Semiconductors (ITRS),Sematech Inc. , *http://public.itrs.net*,2003.

[2] S. H. Gunther, F. Binns, D. M. Carmean, J. C. Hall, "Managing the Impact of Increasing Microprocessor Power Consumption", *Intel Technology Journal*, vol. 5, no. 1, pp. 1-9, 2001.

[3] 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.

[4] P. K. Sahu, S. Chattopadyay, "A Survey on Application Mapping Strategies for Network-on-Chip Design," *Journal of System Architecture, Elsevier*, vol. 59, issue 1, pp. 60-76, 2013.

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

[6] Hotspot 5.01, Thermal Modeling Tool for Integrated Circuits.

http://lava.cs.virginia.edu/HotSpot

[7] 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.

[8] C. Tsai, S. Kang, "Cell-level placement for improving substrate thermal distribution," *Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, IEEE Press, NJ, USA, pp. 253– 266, 2000.

[9] O. Ozturk, M. Kandemir, S. W. Son, "An ILP based Approach to Reducing Energy Consumption in NoC based CMPs," *IEEE International Symposiun on Low Power Electronics and Design (ISLPED)*, pp. 411-414, 2007.

[10] P. Ghosh, A. Sen, A. Hall, "Energy Efficient Application Mapping to NoC Processing Elements Operating at Multiple Voltage Levels," *IEEE International Symposiun on Network-on-Chip (NoCS)*, pp. 80-85, 2009.

[11] J. Huang, C. Buckl, A. Raabe, A. Knool, "Energy-Aware Task Allocation for Network-on-Chip Based Heterogeneous Multiprocessor Systems," *Euromicro International Conference on Parallel, Distributed and Network based Processing (PDP)*, pp. 447-454, 2011.

[12] C. L. Chou, R. Marculescu, "Contention-Aware Application Mapping for Network-on-Chip Communication Architectures," *IEEE International Conference on Computer Design (ICCD)*, pp. 164-169, 2008.

[13] S. Tosun, O. Ozturk, M. Ozen, "An ILP Formulation for Application Mapping onto Network-on-Chips," *International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1-5, 2009.

[14] S. Tosun, "Clustered-based Application Mapping Method for Network-on-Chip," *Journal of Advances in Engineering Software 42 (10)*, pp. 868-874, 2011.

[15] 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.

[16] J. Hu, R. Marculescu, "Energy- and Performance-Aware Mapping for Regular NoC Architectures," *IEEE Trasactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551-562, 2005.

[17] J. Hu, R. Marculescu, "Energy-Aware Mapping for Tile-based NoC Architectures under Performance Constraints," *Proceedings of ASP-DAC Conference*, pp. 233-239, 2003.

[18] 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.

[19] T. Shen, C. H. Chao, Y. K. Lien, 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.

[20] 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.

[21] U. Y. Orgas, R. Marculescu, D. Marculescu, "Variation-Adaptive Feedback Control for Network-on-Chip with Multiple Clock Domains," *Proceedings of DAC*, pp. 614-619, 2008.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[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] Y. Xie, W. L. Hung, "Temperature-Aware Task Allocation and Scheduling for Embeded Multiprocessor System-on-Chip (MPSoC) Design," *Journal of VLSI Signal Processing*, 45, pp. 177-189, 2006.

[38] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, D. Tarjan, “Temperature-Aware Microarchitecture”, *IEEE International Symposiun on Computer Architecture(ISCA)*, pp. 1-12, 2003.

[39] L. Shang, L. S. Peh, A. Kumar, N. K. Jha, “Thermal Modeling Characterization and Management of On-chip Networks, *IEEE/ACM International Symposium on Microarchitecture*, pp. 67-78, 2004.

[40] L. Shang, L. S. Peh, A. Kumar, N. K. Jha, “Temperature-Aware On-chip Networks,” *IEEE Micro*, *IEEE Computer Society*, vol. 26, no. 1, pp. 130-139, 2006.

[41] G. M. Link, N. Vijaykrishnan, “Hotspot Prevention Through Runtime Reconfiguration in Network-on-Chip,” *Proceedings of the Design, Automation and Test in Eorope Conference and Exhibition (DATE)*, vol. 1, pp. 648-649, 2005.

[42] A. K. Coskun, T. S. Rosing, K. A. Whisnant, “Temperature Aware Task Scheduling in MPSoCs,” *In the Proceedings of the Design, Automation and Test in Eorope Conference and Exhibition (DATE)*, pp. 1-6, 2007.

[43] C. A. Quaye, “Thermal-Aware Mapping and Placement for 3-D NoC Design,” *IEEE International Conference on SoC*, pp. 25-28, 2005.

[44] W. Hung, C. A. Quaye, T. Theocharides, Y. Xie, N. Vijaykrishnan, M. J. Irwin, “Thermal-Aware IP Virtualization and Placement for Network-on-Chip Architecture,” *IEEE International Conference on Computer design (ICCD)*, pp. 430-437, 2004.

[45] W. Zhou, Y. Zhang, Z. Mao, “Pareto based Multi-objective Mapping IP Cores onto NoC Architecture”, *IEEE Asia Pacific Conference on Circuits and System (APCCAS)*, pp. 331-334, 2006.

[46] I. Anagnostopoulos, A. Bartzas, D. Soudris, “Application-Specific Temperature Reduction Systematic Methodology for 2D and 3D Network-on-Chip,” *Springer, PATMOS 2009*, pp. 86-95, 2010.

[47] Y. Liu, Y. Ruan, Z. Lai, W. Jing, “Energy and Thermal Aware Mapping for Mesh-based NoC Architectures Using Multi-objective Ant Colony Algorithm,” *IEEE International Conference on Computer Research and Development (ICCRD)*, pp. 407-411, 2011.

[48] A. K. Coskun, T. S. Rosing, K. A. Whisnant, K. C. Gross, “Static and Dynamic Temperature-Aware Scheduling for Multiprocessor SoCs,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 9, pp. 1127-1140, 2008.

[49] H. Sarhan, O. K. Eddash, M. Raymond, A. Wassal, Y. Ismail, “Temperature-Aware Adaptive Task-Mapping Targeting Uniform Thermal Distribution in MPSoC platforms,” *IEEE International Conference on Energy-Aware Computing (ICEAC)*, pp. 1-3, 2010.

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

[51] P. K. Sahu, K. Manna, N. Shah, and S. Chattopadhyay, “Extending Kernighan–Lin partitioning heuristic for application mapping onto Network-on-Chip,” *Journal of System Architecture*, DOI: 10.1016/j.sysarc.2014.04.004, 2014.