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

Pradip Kumar Sahu¹, Kanchan Manna², Tapan Shah³ and Santanu Chattopadhyay⁴
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, $G(\mathcal{C}, \mathcal{E})$ with each vertex $c_i \in \mathcal{C}$ representing a core and the directed edge $e_{ij} \in \mathcal{E}$ representing the communication between the cores $c_i$ and $c_j$. The weight of edge $e_{ij}$, denoted by $\text{comm}_{ij}$, 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 $P(\mathcal{U}, F)$ with each vertex $u_i \in \mathcal{U}$ representing a node in the topology and the directed edge $f_{ij} \in F$ representing a direct communication between the vertices $u_i$ and $u_j$. The weight of the edge $f_{ij}$, denoted as $\text{bw}_{ij}$, represents the bandwidth available across the edge $f_{ij}$.

A mapping of the core graph $G(\mathcal{C}, \mathcal{E})$ onto the topology graph $P(\mathcal{U}, F)$ is defined by the function, $\text{map}: \mathcal{C} \rightarrow \mathcal{U}$, such that, $\forall c_i \in \mathcal{C}, \exists u_i \in \mathcal{U}$ and $\text{map}(c_i) = u_i$.

The function associates core $c_i$ to router $u_i$. Naturally, assuming that at most one core can be attached to each router, mapping is defined only when $|\mathcal{C}| \leq |\mathcal{U}|$. 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 $d^k$, $k = 1, 2, ..., |\mathcal{E}|$. The value of commodity $d^k$, corresponding to the communication between cores $c_i$ and $c_j$ is equal to $\text{comm}_{ij}$, the bandwidth requirement. If $c_i$ is mapped to the router $u_i$ and $c_j$ mapped to $\text{map}(c_i)$, the set of all commodities $D = \{d^k\}$ is defined as follows.

$$D = \{d^k|\text{value}(d^k) = \text{comm}_{ij}, \text{for } k = 1, 2, ..., |\mathcal{E}| \text{ and } e_{ij} \in \mathcal{E}\} \quad (1)$$

Also,

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

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

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

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.

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

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

$$T_c = \sum_{e \in E} \text{value (d^e)} \cdot \text{hopcount (source (d^e), sink (d^e))} \quad (5)$$

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.

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 to generate a few task graphs with 64 cores. By varying bandwidth, number of start nodes and input 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/cm²) [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.

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

Where, $T_{avg}$ = Temperature variance of chip

$$T_{tile_i} = \text{Temperature of } i^{th} \text{ tile inside the chip}$$

$$T_{avg} = \text{Average temperature of chip}$$

$$N = \text{Total number of tiles inside the chip}$$

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

$$T_{peak} = \text{Max}(T_{tile}), \text{ for } i = 1, 2, ..., N$$

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%.

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

- Communication cost = 107888.02
- Temperature variance = 146.83
- Peak Temperature = 462.06°C

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

- Communication cost = 119782.0
- Temperature variance = 110.78
- Peak Temperature = 455.53°C

- $T_{tile} = \text{Temperature of } i^{th} \text{ tile inside the chip}$
- $T_{avg} = \text{Average temperature of chip}$
- $N = \text{Total number of tiles inside the chip}$
- $T_{peak} = \text{Max}(T_{tile}), \text{ for } i = 1, 2, ..., N$
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 subproblems, 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. PLBMAR, 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

<table>
<thead>
<tr>
<th>TGFF Graphs</th>
<th>Strategy-1</th>
<th>Strategy-2</th>
<th>Strategy-3</th>
<th>Constructive Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1(64)</td>
<td>10623.10</td>
<td>126.50</td>
<td>454.14</td>
<td>13280.0</td>
</tr>
<tr>
<td>G2(64)</td>
<td>137389.0</td>
<td>153.47</td>
<td>458.0</td>
<td>158329.0</td>
</tr>
<tr>
<td>G3(64)</td>
<td>143073.0</td>
<td>151.05</td>
<td>458.04</td>
<td>152999.0</td>
</tr>
<tr>
<td>G4(64)</td>
<td>56544.40</td>
<td>138.94</td>
<td>457.80</td>
<td>60123.40</td>
</tr>
<tr>
<td>G5(64)</td>
<td>7912.20</td>
<td>134.93</td>
<td>457.78</td>
<td>8600.08</td>
</tr>
<tr>
<td>G6(64)</td>
<td>55068.30</td>
<td>145.65</td>
<td>460.68</td>
<td>61470.10</td>
</tr>
<tr>
<td>G7(64)</td>
<td>9639.14</td>
<td>173.82</td>
<td>436.72</td>
<td>12553.30</td>
</tr>
</tbody>
</table>

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.

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

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**.
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 finds solutions intermediate to these values. First it selects the core sequence using the procedure \( \text{Find_Best_sequence} \). The number of solutions produced by the algorithm is dependent on the \( \text{Effort} \) value provided by the user. If \( \text{Effort} \) is less, algorithm iterates for less number of times exploring small number of solutions. If \( \text{Effort} \) 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 \( \text{Effort} \) 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 \( \text{Map_Graph} \) calls the function \( \text{Do_Mapping_Cal_Cost} \) 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 \( \text{Find_Mapping} \). Once all such solutions have been generated, \( \text{Do_Mapping_Cal_Cost} \) reports only the non-dominated ones for the perusal of the designer.

The procedure \( \text{Find_Mapping} \) 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 \( N_{seq}[\text{start\\_core}] \) 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 \( \text{Evaluate_Positions} \)) and the router positions giving minimum communication cost are copied into the set \( \text{Min\\_Positions} \). If the set \( \text{Min\\_Positions} \) contains a single entry, the corresponding router position is taken to be the \( \text{Best\\_Position} \) for the core. Otherwise, the procedure \( \text{Predict_Best} \) is called to get the best position. The procedure \( \text{Predict_Best} \) attempts to distinguish between the router positions in \( \text{Min\\_Positions} \) set. To evaluate the fitness

\[
wt = wt + \frac{1}{\text{Effort}+1}
\]
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_Termal_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 (c1, c2) be the edge of G with the highest required bandwidth
Cost1 = \sum_{c_{enighbour}(c_1)}^{c} \text{Bandwidth requirement of } (c_1, c_i)
Cost2 = \sum_{c_{enighbour}(c_2)}^{c} \text{Bandwidth requirement of } (c_2, c_i)
If (Cost1 > Cost2) then Start_Core = c1; Else Start_Core = c2
BT_{start} = 1; Best_cost = 1; flag = Cost_Opt
Do_Mapping_Cal_Cost(G, P, Start_Core, flag, wt)
Find_Core_Sequences(G)
flag = Find_Best_Sequence()
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 Graph 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\text{Start Core})
If (flag = Cost_Opt || flag = Both_Opt)
Cost = Compute_Comm_cost(Mapping, G)
If (flag = Temp_Opt || flag = Both_Opt)
T_{aw} = Find_Termal_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 = 0 /* 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}
End while
Comm_cost = Total communication cost for this mapping
Hotspot tool generates temperature profile using these values and each block to generate the temperature profile. The file generated by HotSpot tool is inspected for parameter values for configuration are mentioned in Table 2. 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, 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.

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>
<thead>
<tr>
<th>Table 2. HotSpot parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chip Specifications</strong></td>
</tr>
<tr>
<td>Chip thickness in meter</td>
</tr>
<tr>
<td>Silicon thermal conductivity in W/(m·K)</td>
</tr>
<tr>
<td>Silicon specific heat in J/(m²·K)</td>
</tr>
<tr>
<td>Temperature threshold for DTM (Kelvin)</td>
</tr>
<tr>
<td><strong>Heat Sink Specifications</strong></td>
</tr>
<tr>
<td>Convection capacitance in J/K</td>
</tr>
<tr>
<td>Convection resistance in K/W</td>
</tr>
<tr>
<td>Heat sink side in meter</td>
</tr>
<tr>
<td>Heat sink thickness in meter</td>
</tr>
<tr>
<td>Heat sink thermal conductivity in W/(m·K)</td>
</tr>
<tr>
<td>Heat sink specific heat in J/(m²·K)</td>
</tr>
<tr>
<td><strong>Heat Spreader Specifications</strong></td>
</tr>
<tr>
<td>Spreader side in meter</td>
</tr>
<tr>
<td>Spreader thickness in meter</td>
</tr>
<tr>
<td>Heat spreader thermal conductivity in W/(m·K)</td>
</tr>
<tr>
<td>Heat spreader specific heat in J/(m²·K)</td>
</tr>
<tr>
<td><strong>Interface Material Specifications</strong></td>
</tr>
<tr>
<td>Interface material thickness in meter</td>
</tr>
<tr>
<td>Interface material thermal conductivity in W/(m·K)</td>
</tr>
<tr>
<td>Interface material specific heat in J/(m²·K)</td>
</tr>
</tbody>
</table>

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.

Fig. 4 shows the solutions generated by HotSpot tool is inspected for communication cost. As shown in Fig. 5, 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, 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.

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)

<table>
<thead>
<tr>
<th>Effort</th>
<th>Tolerances of Communication Cost in %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5%</td>
</tr>
<tr>
<td></td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
<tr>
<td></td>
<td>(Comm. Cost, Temp. Variance)</td>
</tr>
</tbody>
</table>

Fig 3: Communication cost vs. temperature variance of different tolerance values for 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.
Table 4. Results for different TGFF graphs and different Effort values for 20% tolerance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>(7284.27, 134.29, 460.38) (7886.17, 125.81, 457.57)</td>
<td>(6988.64, 170.13, 462.83) (7284.27, 134.29, 460.38)</td>
<td>(7284.27, 134.29, 460.38) (7886.17, 125.81, 457.57)</td>
<td>(6988.64, 170.13, 462.83) (7284.27, 134.29, 460.38)</td>
</tr>
<tr>
<td>G2</td>
<td>(133467.0, 160.73, 459.53) (134613.0, 152.16, 459.26)</td>
<td>(132452.0, 174.83, 459.84) (133467.0, 160.73, 459.53)</td>
<td>(134613.0, 152.16, 459.26) (133467.0, 160.73, 459.53)</td>
<td>(134613.0, 152.16, 459.26) (133467.0, 160.73, 459.53)</td>
</tr>
<tr>
<td>G3</td>
<td>(111572.0, 134.44, 458.39) (119782.0, 110.78, 455.53)</td>
<td>(111387.0, 136.74, 458.47) (119782.0, 110.78, 455.53)</td>
<td>(112572.0, 124.54, 458.19) (119782.0, 110.78, 455.53)</td>
<td>(112572.0, 124.54, 458.19) (119782.0, 110.78, 455.53)</td>
</tr>
<tr>
<td>G4</td>
<td>(54634.80, 173.02, 462.39) (54680.80, 150.33, 462.30)</td>
<td>(54680.80, 173.02, 462.39) (54680.80, 150.33, 462.30)</td>
<td>(54829.80, 144.73, 462.24) (54829.80, 144.73, 462.24)</td>
<td>(54829.80, 144.73, 462.24) (54829.80, 144.73, 462.24)</td>
</tr>
<tr>
<td>G5</td>
<td>(6811.29, 162.49, 461.92) (6939.91, 140.55, 460.34)</td>
<td>(6811.29, 162.49, 461.92) (6939.91, 140.55, 460.34)</td>
<td>(6811.29, 162.49, 461.92) (6939.91, 140.55, 460.34)</td>
<td>(6811.29, 162.49, 461.92) (6939.91, 140.55, 460.34)</td>
</tr>
<tr>
<td>G6</td>
<td>(42695.30, 145.45, 460.33) (43782.10, 132.77, 459.31)</td>
<td>(42695.30, 145.45, 460.33) (43782.10, 132.77, 459.31)</td>
<td>(42695.30, 145.45, 460.33) (43782.10, 132.77, 459.31)</td>
<td>(42695.30, 145.45, 460.33) (43782.10, 132.77, 459.31)</td>
</tr>
<tr>
<td>G7</td>
<td>(7153.55, 120.92, 457.30) (7203.07, 118.88, 457.77)</td>
<td>(7153.55, 120.92, 457.30) (7203.07, 118.88, 457.77)</td>
<td>(7153.55, 120.92, 457.30) (7203.07, 118.88, 457.77)</td>
<td>(7153.55, 120.92, 457.30) (7203.07, 118.88, 457.77)</td>
</tr>
</tbody>
</table>
### Table 5. Optimization results for different graphs with 20% tolerance

<table>
<thead>
<tr>
<th>TGFF Graphs</th>
<th>Comm. Cost Optimization (wt = 0)</th>
<th>Temp. Variance</th>
<th>Comm. Cost</th>
<th>Temp. Variance</th>
<th>Comm. Cost and Temp. Optimization (wt = 0.5)</th>
<th>% of change for wt=1.0 w.r.t. wt=0</th>
<th>% of change for wt=0.5 w.r.t. wt=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>6734.78</td>
<td>184.58</td>
<td>7886.17</td>
<td>125.81</td>
<td>7886.17</td>
<td>17.10%</td>
<td>31.84%</td>
</tr>
<tr>
<td>G2</td>
<td>113653.0</td>
<td>185.38</td>
<td>13373.0</td>
<td>155.15</td>
<td>133085.0</td>
<td>17.69%</td>
<td>16.31%</td>
</tr>
<tr>
<td>G3</td>
<td>109327.0</td>
<td>156.01</td>
<td>12327.4</td>
<td>108.65</td>
<td>119782.0</td>
<td>12.76%</td>
<td>30.36%</td>
</tr>
<tr>
<td>G4</td>
<td>48765.4</td>
<td>174.17</td>
<td>57647.2</td>
<td>122.10</td>
<td>57647.2</td>
<td>18.21%</td>
<td>29.90%</td>
</tr>
<tr>
<td>G5</td>
<td>5933.51</td>
<td>175.55</td>
<td>7101.82</td>
<td>139.06</td>
<td>7153.55</td>
<td>19.69%</td>
<td>20.79%</td>
</tr>
<tr>
<td>G6</td>
<td>42086.6</td>
<td>146.98</td>
<td>48480.1</td>
<td>117.11</td>
<td>44885.6</td>
<td>15.19%</td>
<td>25.85%</td>
</tr>
<tr>
<td>G7</td>
<td>6259.47</td>
<td>179.78</td>
<td>7153.55</td>
<td>120.92</td>
<td>7153.55</td>
<td>14.28%</td>
<td>32.74%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>9207.50</td>
<td>149.66</td>
<td>744.40</td>
<td>165.40</td>
<td>6734.78</td>
</tr>
<tr>
<td>G2</td>
<td>132292.3</td>
<td>185.73</td>
<td>120209.5</td>
<td>159.43</td>
<td>107741.9</td>
</tr>
<tr>
<td>G3</td>
<td>116337.8</td>
<td>143.69</td>
<td>118879.4</td>
<td>193.06</td>
<td>107888.0</td>
</tr>
<tr>
<td>G4</td>
<td>55244.17</td>
<td>123.79</td>
<td>49694.11</td>
<td>155.94</td>
<td>45598.64</td>
</tr>
<tr>
<td>G5</td>
<td>6015.28</td>
<td>161.05</td>
<td>6267.59</td>
<td>174.32</td>
<td>5758.61</td>
</tr>
<tr>
<td>G6</td>
<td>44902.16</td>
<td>171.46</td>
<td>42394.97</td>
<td>165.09</td>
<td>40905.62</td>
</tr>
<tr>
<td>G7</td>
<td>9129.94</td>
<td>166.32</td>
<td>7655.83</td>
<td>201.48</td>
<td>6210.72</td>
</tr>
</tbody>
</table>

Normalized Average: 1.214 1.261 1.110 1.404 1.0 1.304 1.176 1.024 1.198 1.0
7. REFERENCES


