Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Energy-efficient task-resource co-allocation and heterogeneous multi-core NoC design in dark silicon era

Md Farhadur Reza [a,*], Dan Zhao [b], Magdy Bayoumi [c]

[a] School of Computer Science and Mathematics, University of Central Missouri, 108 W South St, Warrensburg, MO 64093, USA
[b] Department of Computer Science, Old Dominion University, 3300 Engineering & Computational Sciences Building, Norfolk, VA 23529, USA
[c] Department of Electrical & Computer Engineering, University of Louisiana at Lafayette, 131 Rex Street, Lafayette, LA 70504, USA

ARTICLE INFO

Keywords:
Network-on-chip (NoC)
Dark silicon
Mapping and configuration
Task-resource co-allocation
Dynamic mapping
Energy
Hotspots
Power budget
Multi/many-core

ABSTRACT

To address power limitation issues in dark silicon era for multi-core systems-on-chip and chip multiprocessors, run time task-resource and voltage co-allocation with reconfigurable network-on-chip (NoC) framework for energy efficiency (higher performance/watt) is proposed in this work. Distributed resource managers strategy dynamically reconfigures the voltage/frequency-levels of the NoC links and routers and dynamically power-gates the resources depending on the traffic demands and utilization of the resources. A mapping heuristic, namely *MinEnergy*, has been proposed to minimize overall chip power and energy hotspots in large-scale NoC. We have formulated the mapping and configuration problem into a linear optimization model for the optimal solution and implemented a state-of-the-art *Minimum-Path* contiguous mapping for comparisons. Simulations are carried out under real-world benchmarks and platforms to demonstrate the effectiveness and efficiency of the proposed schemes and results show that the energy, power, and performance of the proposed dynamic mapping and configuration solution are significantly better (more than 50%) than those of *minimum-path* mapping solution, while the energy and power consumption of the proposed solution are more than 90% close to the optimal solution.

1. Introduction

To achieve more substantial energy-performance efficiency, multi-core designs shift to many core integration with hundreds and even a thousand of cores to cope with Moore's Law scaling. The industries and academicians are working to integrate many cores on a chip to meet the high-performance demands of the current and emerging applications, such as machine learning, big data, image processing, and scientific-computing. A single chip server with many cores can replace a rack of servers in a data center or can even replace a whole data center [1]. For example, Cerebras have developed a single chip with 400K cores using 1.2 trillion transistors [2], and University of California Davis have developed a 1000-core chip (with IBM), which has a maximum computation rate of 1.78 trillion instructions per second and contains 621 million transistors. The companies, such as Google and Facebook, are working with the chip companies (e.g., Intel) to get a small-scale on-chip server solution to replace their big data centers. For example, Facebook with the help of Intel has developed system-on-chip compute server called Yosemite that holds four SoC processor cards and provides the flexibility and power efficiency for scale-out data centers [3]. This solution has reduced Facebook's costs tremendously in most of the areas, such as, operation and maintenance costs, power consumption, and warehouse space. Other companies (e.g., AMD, Amazon) have also taken initiatives towards on-chip servers. In multi/many-core systems, all the cores can run in parallel to run different applications. To meet user and application demands, both processor (core) speed and count on a chip are increasing, and the growing rate of processor speed and count on a single chip is putting more pressure on the power and thermal budgets of a chip.

With the advancement and miniaturization of transistor technology, hundreds to thousands of cores can be integrated on a single chip [1]. But power consumption of transistor does not decrease in the same way as the transistor size decreases due to the problem with energy scaling of transistor technology, which results in Dennard scaling failure [4,5]. As shown in [4,6], across two process generations, we can either increase core count by 2x, or frequency by 2x due to power budget limitation. The remaining potential results in either dark or dim silicon depending on the preference for frequency (and voltage) versus core count. Dark silicon refers to the fully deactivated resources (zero supply voltage), where dim silicon is the resources with reduced frequency/voltage (greater than zero voltage). The dark silicon problem

* Corresponding author.
E-mail addresses: reza@ucmo.edu (M.F. Reza), zhao@cs.odu.edu (D. Zhao), magdy.bayoumi@lousiana.edu (M. Bayoumi).

https://doi.org/10.1016/j.micpro.2021.104055
Received 11 June 2020; Received in revised form 5 January 2021; Accepted 10 February 2021

arises because of the increasing number of cores (transistors) integration in a chip [7]. Besides power limitation, a component of the chip cannot exceed a manufactured thermal limitation at any instant of time, as a thermal (energy) hotspot increases several failures (e.g., electromigration), and can cause permanent damage to the chip, e.g., burn the chip. High temperature increases leakage power, which is the major power contributor in deep-submicron technologies. Power and thermal limitations in the dark silicon era have been discussed in [4,6,7].

One of the challenging research problems in multi/many-core computing is the mapping of resource demanding tasks onto various homogeneous/ heterogeneous processor tiles to meet specific performance objective. The mapping optimization in heterogeneous NoC faces new challenges of very limited on-chip resource usage including tight power and thermal budgets. It becomes even more critical when the core count keeps on increasing with ever shrinking chip size.

The contributions of this work are outlined below:

- *Dark Silicon Challenges for Task-Resource Co-Allocation:* Design challenges of multi/many-core NoC in dark silicon are discussed in Section 3. Application-architecture mapping issues in many-core NoC in dark silicon are presented in Section 4.
- *Linear Programming Optimization Model:* Mapping and configuration problem considering dark silicon challenges, chip constraints and application demands is formulated using linear programming (LP) in Section 5 to get the optimal solution. A heterogeneous NoC is established with varying voltage/frequency (V/F) levels at the routers and links of the NoC to reduce the power consumption. The optimal solution from LP is used to find the near-optimal property of our proposed run-time mapping solution.
- *Run-time Resource Optimization in Dark Silicon:* Power and hotspots aware task mapping and NoC resource configuration are dynamically co-designed for energy orchestration for the first time in this work (to the best of our knowledge) in Section 7. Varying voltages are applied at the routers and links of NoC to reduce the energy consumption. By supporting the resource (core/router/link) deactivation feature, it further diminishes the power dissipation. Energy hotspots are minimized by avoiding overloading resources to meet the tile-level chip power budget. As an LP optimal solution has a high time complexity, a fast heuristic is proposed (in Section 7) to get the task mapping and resource reconfiguration solution for large NoCs and applications in a quick time (at run-time).
- *Distributed Resource Management:* Distributed strategy for NoC resource management while running applications on NoC based large-scale systems has been presented in this work. Distributed cluster managers are used for resource status monitoring and activation/deactivation of resources in a cluster while a global manager is used for mapping and configuration decisions.

Related work is discussed in Section 2.

2. Related work

Task-Resource allocation and NoC configuration are driven by various energy and performance goals, such as minimizing energy consumption, reducing delay and improving throughput of applications, and meeting QoS (quality of service) of real time jobs. Task-resource allocation can be carried out at design time (static) or run-time (dynamic). Static mapping techniques for multiprocessor NoCs are to find dedicated allocation of tasks at design time along with architectural design and configuration [8–12]. Most of the static mapping solutions try to map the closely communicating tasks to neighbor cores to improve the communication and locality of data accesses [8–11,13,14]. Some works minimize the resource contention, which arises in overcrowded computation and communication region, e.g., [15–17]. In [8], communication traffic is split onto multiple paths to improve bandwidth efficiency during task mapping. Routing flexibility is exploited in [10] to expand the mapping solution space, and a branch-and-bound algorithm has been proposed to solve the mapping problem wherein the time complexity increases exponentially with the increase in NoC size. It was proposed in [14] to map the closely communicating threads to neighboring cores to improve locality of data accesses. Although it effectively reduces overall communication volume, resource contention can be introduced in communication and computation intensive regions. A multi-application mapping method proposed in [18] attempts to find the optimal region for each application using maximal empty rectangle technique.

Requirements for generalized chip that can meet the demands of multiple unknown applications at run-time have been increasing, and our work is motivated by that requirement. If a new application arrives in order to run on the current chip, chip resources should be configurable depending on the traffic demands of the incoming application. That is why the need for dynamic mapping and reconfiguration arises, and our work is motivated by that requirement. For adaptive systems with dynamic workloads, run time mapping techniques are required to accurately addressing the workload characteristics [13,19–22]. Like static mappings, run-time mapping in [19] also proposed contiguous neighborhood allocation algorithm that maps the communicating tasks in a close neighborhood, and tried to have contiguous mapping regions (by reducing unmapped fragmented regions between mapped regions). [20] proposes hill-climbing heuristic to find the start node with the required number of contiguous nodes (around the start node) to meet demands of the corresponding application. After selecting the first node, the proposed approach map the tasks of the application contiguously. An incremental run-time application mapping algorithm in 3D-NoC was proposed in [21], which maps tasks to reduce communication cost and follows region-based application mapping to minimize fragmentation of the tasks of an application. A run-time application mapping approach in homogeneous NoC with multiple voltage levels was proposed in [13], where the centralized agent approach is used and data network is separated from control network for efficient data delivery. [13] work was extended to [22] by integrating dynamic behavior (communication rate and energy cost) of applications. [22] minimizes both communication load and contention in the network. Various heuristics were investigated for run-time application mapping in [23], where the main aim is to minimize the load (and congestion) on the NoC links. Dynamic mapping in [24] proactively select a square region required for incoming application to reduce congestion and dispersion in the mapped neighborhood. Run-time mapping in dark silicon to increase power budget for activating more resources were presented in [25,26]. Thermal constrained static resource management in dark silicon is proposed in [27]. [28] summarizes the initiatives to exploit dark silicon to mitigate power and temperature density problems in many-core systems. Various user applications are considered in [12,29] such that the tasks are mapped to either meet the worst case requirement or dynamically switch between different user-application requirements. In this work, we address the dynamic task mapping problem in reconfigurable heterogeneous many-core NoC, aiming to lower the overall energy and hotspots (i.e., total workload) subject to computation, communication, deadline, and power budgets constraints. NoC links and routers capacity are minimally configured with V/F-levels to fulfill communication interactions.

Centralized manager approach presented in some work [13,20] is not feasible for hundreds of nodes based NoC because of the hotspot and communication delay problems. A single centralized becomes a communication hotspot as many communications are going through the controller. Also remote nodes face significant communication delay for quick and effective monitoring and reconfiguration of the NoC resources, as NoC statistics collection and configuration decision suffer delay for its effective application in real-time systems. Distributed cluster-agent based run-time mapping approach has been presented in [30], where global agent communicates with the distributed cluster agents and then finally take the decisions onto which cluster the

application should be mapped. Run-time agent-based distributed application mapping for on-chip communication has been presented in [30]. Distributed resource management for on-chip many-core systems was discussed in [31]. Therefore, in this work, distributed managers based mapping and configuration techniques in large-scale systems have been proposed. The uniqueness of this work is that this work focuses to reduce both overall energy and hotspots at the same time by dynamically mapping the tasks and configuring the NoC resources with the help of distributed managers while satisfying the chip and application constraints.

3. Design challenges of multi/many-core NoC in dark silicon

NoC, as a communication network, consumes a significant percentage of on-chip power. For example, on-chip network consumes 36% of the total chip power in 16-tile MIT Raw [32,33]. The NoC power consumption has been increasing with the increase in core integration on the chip. But power is a very valuable thing in the current world, and every device has a limitation on its power budget. Therefore, power consumption becomes a critical issue in many-core on-chip systems. For example, 21% and 50% of the chip resources may have to be dark at 22 nm and 8 nm process technology, respectively [7]. This dark silicon concept becomes a major solution for current power limitation in many-core chip [4,7,34]. As a significant power source, NoC resources can be configured dynamically to decrease pressure on power budget (by using the least capacity to meet traffic demands). Besides power limitation, a component of the chip cannot exceed a manufactured thermal limitation at any instant of time. Temperature of an active node[1] is affected by its neighbor nodes temperature. If a neighbor node is dissipating heat (because of higher load), then some percentage of that heat also reaches its neighborhood. To reduce the thermal impact of an active node to another active node, we mixture the active and dark nodes together (as a dark node does not have heat impact on the active nodes) instead of completely separating dark nodes from active regions (as in [25,35]). This distribution of active and dark nodes helps us to run more tasks with the same thermal design power (TDP) budget [36]. To reduce energy hotspots in the many-core heterogeneous NoC, dark nodes need to be distributed uniformly throughout the network to reduce the thermal impact of nodes on each other. [37] proposed a new thermal metric, thermal safe power (TSP), to better address the dark silicon issue in many-core systems compared to TDP. TDP is a fixed constant value, which does not fully exploit the available power budget in the chip like TSP. TSP varies with the number of active cores (and dark cores). Temperature, reliability, and variability challenges in the dark silicon era have been presented in [28,38]. Instead of a single power or thermal metric, this work uses the combination of both overall chip power and tile wise power budgets for resource allocation decisions to address dark silicon issues in NoC based multi/many-core systems.

4. Task-resource co-allocation in multi/many-core NoC in dark silicon

With the ongoing many-core revolution, hundreds to thousands of computational threads per chip are achievable. However, while transistors keep shrinking, current leakage poses greater challenges, and consequently power density increases, causing the failure of Dennard Scaling [5]. When facing a severe power wall, not all cores can be powered on at full speed simultaneously, leaving a large amount of silicon inactive or operating at lower frequency, the so called dark/dim silicon [4]. To effectively leverage application and task parallelism in dark/dim silicon era, it is essential to investigate run-time task mapping under the critical power and thermal constraints.

[1] Node is used indicate both router and core at a tile.

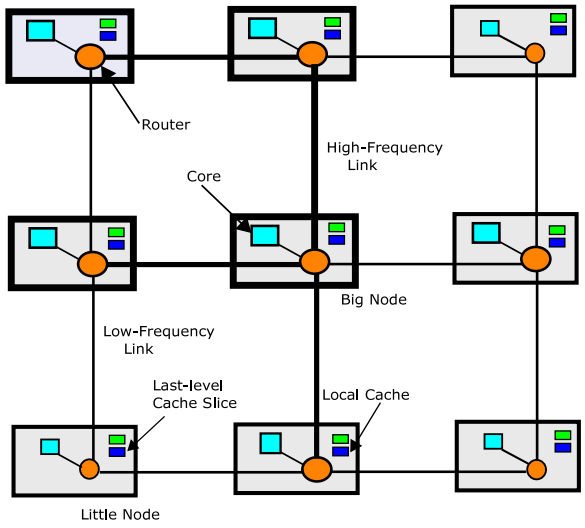


Fig. 1. Heterogeneous Node and Link Configuration in 3 × 3 NoC.

Most of the existing mapping approaches map the closely communicating tasks to neighbor cores to improve the communication and locality of data accesses [8,19]. A significant bottleneck of contiguous mapping is the formation of energy hotspots, which is the major problem of dark silicon, due to mapping in close neighborhood. Some hotspots reduction mapping solutions [39] reduce the hotspots but increase the overall communication load for remote mapping of communicating tasks. Task mapping in dark silicon is different as the task mapper needs to decide whether to run the task on the already active core or it needs to wake-up a deactivated core to map the task (considering energy consumption penalty to wake-up a core). The challenge of task mapper is to choose the set of active cores for running applications within the power budgets. Existing mapping solutions with the exception of a few [25,27] do not consider dark silicon during task-resource allocation. This work activates and deactivates the resources and minimizes energy hotspots in many-core silicon during task-resource co-allocation.

A multi-core chip is illustrated in Fig. 1 where heterogeneous processor tiles are placed in a 2D grid. Each tile contains a processor (with distinct computational power), its local cache, a slice of the shared last-level cache, and a router (configured with different communication capacity) for data/control transmission between the tiles via varying load links. The routers are interconnected to form a NoC. To cope with power consumption limitation, a heterogeneous multi/many-core architecture is envisioned to offer a feasible approach that embraces high-performance 'Big' cores (cores with higher computational capacity) for computation intensive applications and low-power 'Little' cores (cores with lower computational capacity) for majority workloads execution.

A multi/many-core chip may run multiple applications simultaneously depending on the number of tasks in the applications and number of cores in the chip. To fully utilize the computation power of the cores, an application can be partitioned into a number of tasks to be simultaneously processed on different processor tiles. Meanwhile, multiple tasks with diverse computation workloads from different applications may share the same tile under the restricted on-chip resource budget. Applications may not be known in advance, and scenario of an application may even change at run-time. That is why, the need for dynamic mapping and configuration arises. Though application scenario can change dynamically, we still need to assume applications trends and characteristics to design and configure our chip. So, we assume each application has been appropriately partitioned into a set of tasks that can be executed concurrently. While task partitioning is beyond the scope of this work, the interested readers are referred

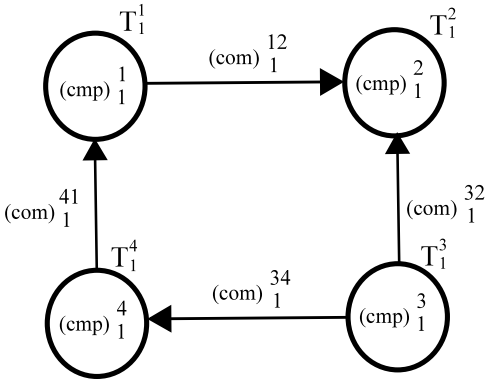


Fig. 2. An example of application task graph G.

to [40] for more details. Data are transferred between communication dependent tasks via NoC to fulfill computation of the applications. Our run-time mapping framework maps application(s) dynamically depending on their computation and communication requirements. To cope with the heterogeneity of traffic workloads, NoC is configured heterogeneously using varying V/F-levels for energy efficiency while allocating demanded communication capacity under resource sharing. In this work, the task mapping and NoC configuration are co-designed for energy orchestration of a heterogeneous NoC. We name such a problem task-resource co-allocation. Task-resource co-allocation, like bin packing problem, is an NP-hard problem [41], where time complexity increases very quickly as the size of the problem grows.

5. Linear programming problem formulation

A set of K applications can arrive at the system for mapping. An application A_k can have M variable tasks, and mth task of A_k is represented by T_k^m. Computation demand of T_k^m is $(cmp)_k^m$, and communication requirement between tasks m and n is $(com)_k^{mn}$, as shown in Fig. 2. Our target is to efficiently map the incoming tasks of the application(s) onto the given chip plane with N processor tiles connected into a $\sqrt{N} \times \sqrt{N}$ 2D-mesh (as shown in Fig. 1).

Let $(\alpha_k^m)^i$ be a binary variable to indicate if a task T_k^m is mapped to node i, i.e.,

$$(\alpha_k^m)^i = \begin{cases} 1, \text{if task } T_k^m \text{ is mapped to Node } i, \\ 0, \text{otherwise.} \end{cases} \tag{1}$$

$(\alpha_k^m)^i$ and B_h are to be determined in order to achieve the mapping and configuration optimization goal.

5.1. Resource activation–deactivation

In our proposed solution, we activate and deactivate both NoC (router and link) and core resources. Let C_i^{ON} be a binary variable to indicate if a core (node) i is active, i.e.,

$$C_i^{ON} = \begin{cases} 1, \text{if a task is mapped to Core } i, \\ 0, \text{otherwise.} \end{cases} \tag{2}$$

Let L_{ij}^{ON} be a binary variable to indicate if a link l_{ij} between node i and j is active, i.e.,

$$L_{ij}^{ON} = \begin{cases} 1, \text{if link } l_{ij} \text{ is carrying any traffic,} \\ 0, \text{otherwise.} \end{cases} \tag{3}$$

Let R_i^{ON} be a binary variable to indicate if a router (node) i is active. Router i can be active if a task is mapped to the local core i or the links attached to the router i are carrying traffic, i.e.,

$$R_i^{ON} = \begin{cases} 1, \text{if } (C_i^{ON} + L_{ij}^{ON}) >= 1, \\ 0, \text{otherwise.} \end{cases} \tag{4}$$

5.2. Link communication constraint and voltage/frequency configuration

After mapping of tasks, a communication flow from node p to node q go through a set of links based on the routing algorithm, and $Path_{pq}$ contains all those links. Total traffic on a link l_{ij} can be calculated by summing up all the traffic that goes through that link, which is found by searching though all the $Path_{pq}$. All the traffic go through a link l_{ij} must not exceed the maximum communication capacity BW_{ij} possible on that link, i.e.,

$$bw_{ij} = \Big(\sum_{\substack{1 \le p,q \le N \\ 1 \le m,n \le M \\ 1 \le k \le K \\ \exists l_{ij} \in Path_{pq}}} (com)_k^{mn} \cdot (\alpha_k^m)^p \cdot (\alpha_k^n)^q \Big) \le BW_{ij}, \forall 1 \le i,j \le N. \tag{5}$$

The V/F-level of a link is configured based on the required traffic demands in Eq. (5) using a load and V/F mapping table.

5.3. Node computation-communication constraints and voltage/frequency configuration

For a router i, we sum up all the traffic on the links attached to that router using Eq. (5). Based on the total communication demands at a router, we set the V/F-level of the router using a load and V/F mapping table.

For computation, let P_i^{cap} is the computational capacity of core (node) i. Then the sum of the computation demand of all the tasks assigned to core i cannot exceed P_i^{cap}, i.e.,

$$\rho_i = \sum_{\substack{1 \le k \le K \\ 1 \le m \le M}} (cmp)_k^m \cdot (\alpha_k^m)^i \le P_i^{cap}, \forall 1 \le i \le N. \tag{6}$$

5.4. Chip-wise power budget constraint

Dynamic power consumption of a core pw_i^{core} is calculated by multiplying the frequency (F), capacitance, and square of the voltage level (V) of the core i. Power consumption pw_{ij}^{link} on a link l_{ij} is measured from the link load and frequency of communication. Router power consumption pw_i^{router} depends on the buffer, crossbar, switch allocator, and V/F of the router i. Power is also dissipated while activating and deactivating the components. Also, leakage power is consumed on idle components. The overall power consumption of the chip components PW_{chip} should not exceed the power budget of a chip, denoted as PW_{budget}, i.e.,

$$\begin{aligned} PW_{chip} = \Big(& \sum_{1 \le i,j \le N} pw_{ij}^{link} + \sum_{1 \le i \le N} (pw_i^{router} + pw_i^{core}) + \\ & pw_{activation}^{resource} + pw_{deactivation}^{resource} + pw_{leakage}^{resource} \Big) \\ & \le PW_{budget}, \forall 1 \le i,j \le N. \end{aligned} \tag{7}$$

5.5. Tile-level power budget constraint

As temperature increases proportionally with the increase in power dissipation, we address the thermal issue by using a constraint on maximum power dissipation from a tile of the chip during mapping, where a tile contains core(s), a router and links associated with the router. As high thermal dissipation (because of high power dissipation) of a single chip component can affect the whole chip (e.g., burn), the mapping solution needs to ensure that it is not overloading a single tile. Power consumption at a tile is calculated by summing up the power of the router and associated core and links attached to that tile. Power dissipation of a tile i cannot exceed maximum tile-level power budget of a chip, denoted as TPW_{budget}, i.e.,

$$\begin{aligned} PW_i = \Big(& \sum_{1 \le i,j \le N} (pw_{ij}^{link} + pw_i^{router} + pw_i^{core} + pw_{activation}^{i,ij} \\ & + pw_{deactivation}^{i,ij} + pw_{leakage}^{i,ij}) \Big) \le TPW_{budget}, \forall 1 \le i,j \le N. \end{aligned} \tag{8}$$

TPW_{budget} is empirically estimated depending on the chip power budget, application demands, and mapping solution.

5.6. Task deadline constraint

If a task m has $((T_{cmp})_k^m)^p$ execution time at processor p and $((T_{com})_k^{mn})^{pq}$ communication time with task neighbor n that is mapped to processor q and given a deadline for task m is D_k^m, then deadline constraint of task m, which starts at $(st_{cmp})_k^m$ time can be presented by the following equation:

$$\sum_{\substack{1\le p\le N\\ 1\le m\le M\\ 1\le k\le K}} ((T_{cmp})_k^m)^p \cdot (\alpha_k^m)^p + \sum_{\substack{1\le p,q\le N\\ 1\le m,n\le M\\ 1\le k\le K\\ \exists l_{mn}\in E}} ((T_{com})_k^{mn})^{pq} \cdot (\alpha_k^m)^p \cdot (\alpha_k^n)^q + \sum_{\substack{1\le p\le N\\ 1\le m\le M\\ 1\le k\le K}} (st_{cmp})_k^m \le D_k^m. \tag{9}$$

5.7. Linear programming objective function

The objective of task-resource co-allocation and optimization is to map the tasks of the application(s) into a multi/many-core chip plane with N processor tiles, while meeting the chip-level power budget, tile-level power budget, computation and communication capacity of NoC, and task deadline constraints. Our objective is to minimize the total energy dissipation of all the nodes and links in the chip, i.e.,

$$Minimize : sum(e_{cmp} \cdot \rho_i + e_{com} \cdot \sum_{\forall j \in l_{ij}} bw_{ij}), 1 \le i, j \le N, \tag{10}$$

subject to the constraints given in Eqs. (5), (6), (7), (8), (9). Energy is determined by multiplying load with energy dissipation per load unit. Total load at a tile i is calculated by summing up its computation load and all the communication loads going through that tile (router and core). e_{cmp} and e_{com} are the energy coefficient of computation and communication loads at tile i, respectively. The unit of computation is operations per second and the unit of communication is bit per second. In 22 nm technology, e_{cmp} and e_{com} are around 45 *picojoule* (pJ) per computational operation (we assume 8-bit per computational operation for simplicity) and 65 pJ per communicating bit, respectively [42]. The above objective function along with the communication, computation, deadline, and power constraints form an LP model. Its solution suggests a task mapping strategy, activation/deactivation of NoC resources, and voltage/frequency configurations of the NoC resources, aiming at minimizing energy chip-wide while meeting computation and communication requirements under strict power budgets.

6. Design-time and run-time task mapping and NoC configuration

A many-core system consists of heterogeneous cores (like big/ little cores in [43]) and routers. An application has heterogeneous computation and communication demands. Multiple applications with heterogeneous demands needs to run in a multi/many-core single chip system. Based on the optimization objectives (low power, high performance, hotspots minimization), the system and network need to be configured to produce an energy-efficient solution. Design-time NoC configuration consists of several parameters, including: task-to-node assignment, setting link-width and flit-width, number of cores per router, and setting router design (e.g., number of buffers and virtual channels). Design-time task-resource co-allocation needs to consider the chip constraints in terms of chip-level power budget, tile-level power budget, and computation and communication capacity. Mapping needs to satisfy both computation and communication demands of the tasks. Design-time or static mapping techniques for many-core NoCs are to find dedicated allocation of tasks at design-time along with architectural design and configuration.

Besides the design-time challenges, the run-time optimization needs to configure the system and network dynamically under different workloads of different applications. An application with unknown demands can arrive in the system and/or an application demand may change at run-time. Utilization of heterogeneous resources of multi/many-core NoC-based system needs to be monitored for efficient allocation of resources for incoming and (already) running tasks and applications. A task may need to be migrated from a node to another node to improve the energy-efficiency and/or performance of the system. The system needs to address the application(s) demands, and configure the system for efficient solution. The configuration includes: task-to-node assignment, dynamic voltage and frequency scaling of nodes and links, power-gating, and routing algorithm (to route packets from sources to destinations).

We discuss resource status monitoring and collection, distributed resource management, NoC resource reconfiguration, routing algorithm, and task migration strategies for run-time optimization and configuration in multi/many-core systems.

6.1. Resource status monitoring and collection

Because of heterogeneous application demands, run-time resource allocation and configuration needs to determine the required resources in the next time interval. A resource manager monitors and configures the resources (router and link) based on its managed NoC region: a cluster of nodes or a node. Resource managers transfer the resources status of the corresponding NoC region to the global manager via control network in NoC. A resource manager collects following statistics in a fixed interval, such as every 100 cycles, for the corresponding NoC region: core-task mappings status, resources activation/deactivation status, capacity utilization of resources, and tasks start-finish status. These statistics are used for mapping decision of incoming tasks and configuration of NoC resources in the next interval.

6.2. Distributed resource management

To address power-thermal-dark silicon issues in large-scale server-on-chip, run time task-resource co-allocation with reconfigurable network-on-chip (NoC) framework is needed to accurately address the workload characteristics. Centralized manager approach, presented in some works [13,20,23], is not feasible for hundreds of nodes based NoC. In centralized manager approach, every node has to communicate with the centralized agent for local NoC configuration decisions. This increases traffic to the centralized controller significantly and so this increases the chance of hotspots creation at the controller and/or around the regions of the controller. Furthermore, in the centralized manager approach, mapping and configuration decision communication delay (from resource manager to the resource) increases, which is not effective for run-time solutions in many-core systems. Therefore, distributed agent-based mapping and configuration approach is needed for scalable run-time solutions in many-core systems. We assume many heterogeneous big and little cores are distributed throughout the chip. Resource monitoring and controlling are done by the distributed run-time manager in every cluster, where the chip is partitioned into a number of clusters of uniform size. Clustering, like mapping problem, is an NP-hard problem [41]. The decisions on the number of clusters and size of a cluster depend on the number of nodes in NoC. NoC is uniformly divided into clusters of square or rectangular shape depending on the number of nodes in NoC, for example, 64-router NoC is divided into 4 clusters of $4X4$ routers. The algorithm tries to achieve square shape cluster (e.g., $4X4$, $6X6$). We focuses to achieve clusters with a minimum number of nodes. Minimum number is used to reduce the overhead of the managers by reducing the number of clusters.

A cluster manager monitors and configures the resources within that cluster. Cluster managers transfer the resources status of the corresponding cluster to the global manager via control network in NoC. A cluster manager collects statistics in a fixed interval for the corresponding cluster. These statistics are used for mapping decision of incoming tasks, power-gating of idle resources, and activation of the required resources in the next interval. The distributed managers of the clusters transfer the resources status of the clusters to the global

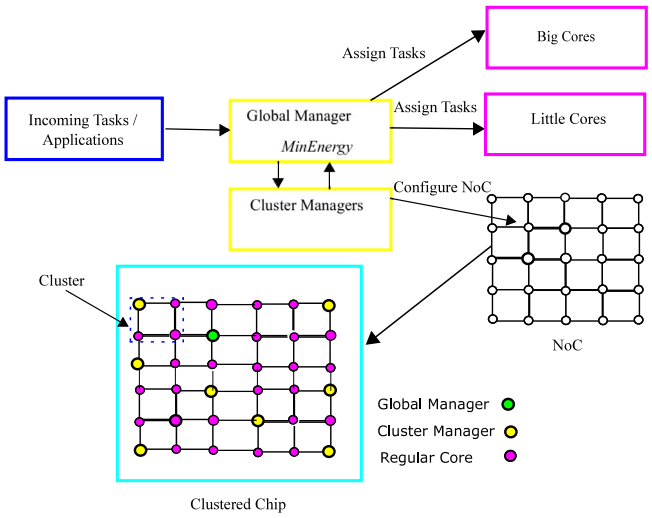


Fig. 3. Distributed Resource Management in NoC.

manager via control network in NoC. The global system manager takes the mapping and configuration decision. The proposed distributed resource management in many-core NoC is presented in Fig. 3. The overall algorithmic process for cluster management and NoC resource configurations is shown in Algorithm 1.

Algorithm 1: Clustering and NoC Resource Management Process

input : Arrived Application TGs, NoC Topology Graph
output : Clustering, NoC Resource Configuration
Step-1: Uniformly divide the NoC into clusters of square or rectangular shape, depending on the number of nodes in NoC
Step-2: Assign a node as a cluster manager for each NoC cluster
Step-3: Assign a node as a global manager for whole NoC
Step-4: Cluster managers monitor the NoC resources and send the statistics to the global manager
Step-5: The global manager maps the incoming task(s)
Step-6: The global manager produces the resource configuration decisions and sends the decisions to the cluster managers
Step-7: Cluster managers configure the NoC resources
Step-8: Repeat step-5 to step-7 till all the arrived applications and/or tasks are mapped;
return mapping and configuration;

The global manager maps the tasks of applications to the cores using *MinEnergy* mapping algorithm (which will be discussed soon). When an application has been finished with its execution and leave the system, capacities of the NoC resources are updated by the system manager accordingly. After the mapping of all the tasks in the queue, the cluster managers deactivate the NoC resource(s) that has been idle (not used by any tasks) for a certain time. A core remains deactivated if no task is mapped to that core. A link is deactivated if that link is not carrying any communication traffic. We shut-down a router only when all the links and the core attached to that router are not being utilized. Router deactivation is very effective in many-core NoC, as router is the major contributor to on-chip network power consumption, e.g., router consumes 36% of total on-chip network power in 16-tile MIT Raw [32]. A NoC router is power-gated if all the links and the core connected to that router are not carrying any traffic and deactivated. For deactivation of a resource component, corresponding cluster manager requests global manager for switching-off that resource into power-off state. The global manager may not approve the request if the number of deactivation request for the same component exceeds a threshold to reduce fluctuating activation–deactivation component status. Thresholds are set based on the traffic demands of the applications. The global manager activates a deactivated component if that component is needed for computation and/or communication demand. This activation is done by forwarding a wake-up signal to the deactivated component.

The clustering concept reduces circuit complexity by controlling all the resources of a cluster by using one power source (per cluster). This cluster strategy further reduces power consumption by deactivating the whole cluster while no task is mapped to this cluster. Our target is to satisfy the requirements of all the applications with lower number of active clusters. We try to increase the utilization of NoC and core resources within a cluster, while trying to reduce the number of low utilized clusters. We also deactivate unused resources within a cluster to drop the overall utilization of the cluster below the pre-defined utilization threshold, 70%. Switching on-off resources by waking-up or turning-off signals have some overhead and this process takes some communication time and this time is considered in task completion time calculation to avoid violating the task deadline constraint.

A flow of the dynamic mapping and configuration solution is presented in Fig. 4. The mapping and configuration solution starts when a new task/application arrives for running or the demand of the existing task/application changes. More than one task or application can arrive in the system with new demands for mapping. The global manager uses current NoC statistics, task demands, and *MinEnergy* mapping algorithm (in Section 7) for mapping decisions of incoming tasks and configurations of NoC resources. The global system manager collects NoC statistics with the help of distributed cluster managers. Each mapping and configuration decision needs to satisfy power, time, and computation/communication capacity constraints (mentioned in Section 5). Distributed managers execute the decisions of the global manager. V/F-levels (including activation) of the cores and routers/links are configured due to change in computation and communication demands, respectively. Idle resources are deactivated as they are not needed for running tasks/applications in the system (which may need to be activated later). This flow repeats for next incoming tasks/applications with new demands.

6.3. NoC resource reconfiguration and routing algorithm

Our proposed mapping solution is application agnostic. During run-time, depending on the mapping solutions, the system manager activates the required NoC links and routers. This application agnostic assumption is considered to make the problem more realistic to adapt our solution to unknown applications because of the new incoming application and/or changing nature of an application. During run-time, our solution maps the application(s) to the chip, and reconfigures the NoC resources.

A link in a NoC can carry traffic of different tasks and applications. At run-time, based on the communication demands on a link, voltage-level of a link is configured (reduced or increased). Reduction in the power consumption of the links allows us to run more resources with the same design power constraint. Thus, run-time link voltage configuration of NoC resources helps to solve the on-chip power design limitation to some extent in on-chip networks.

Shortest-path algorithm is used to forward traffic from a source to a destination so that performance does not drop while minimizing energy dissipation. Another alternative is to bypass (by using misrouting) the deactivated resources to reach the destination router (which increases the communication length). This approach can be implemented by deactivating the NoC router and link resources if no task is mapped to the corresponding core. In that alternative approach, activated resources will have high amount of traffic in some activated links and routers where many resources are under-utilized or deactivated. This longer routing path away from the deactivated resource(s) can create high contention in the activated NoC resources, and so this solution can increase network latency and can reduce network throughput. To avoid longer-path routing and high contention, we activate the NoC router and link resources if the corresponding resources are needed by the communicating tasks. Our main objective is not to detour the

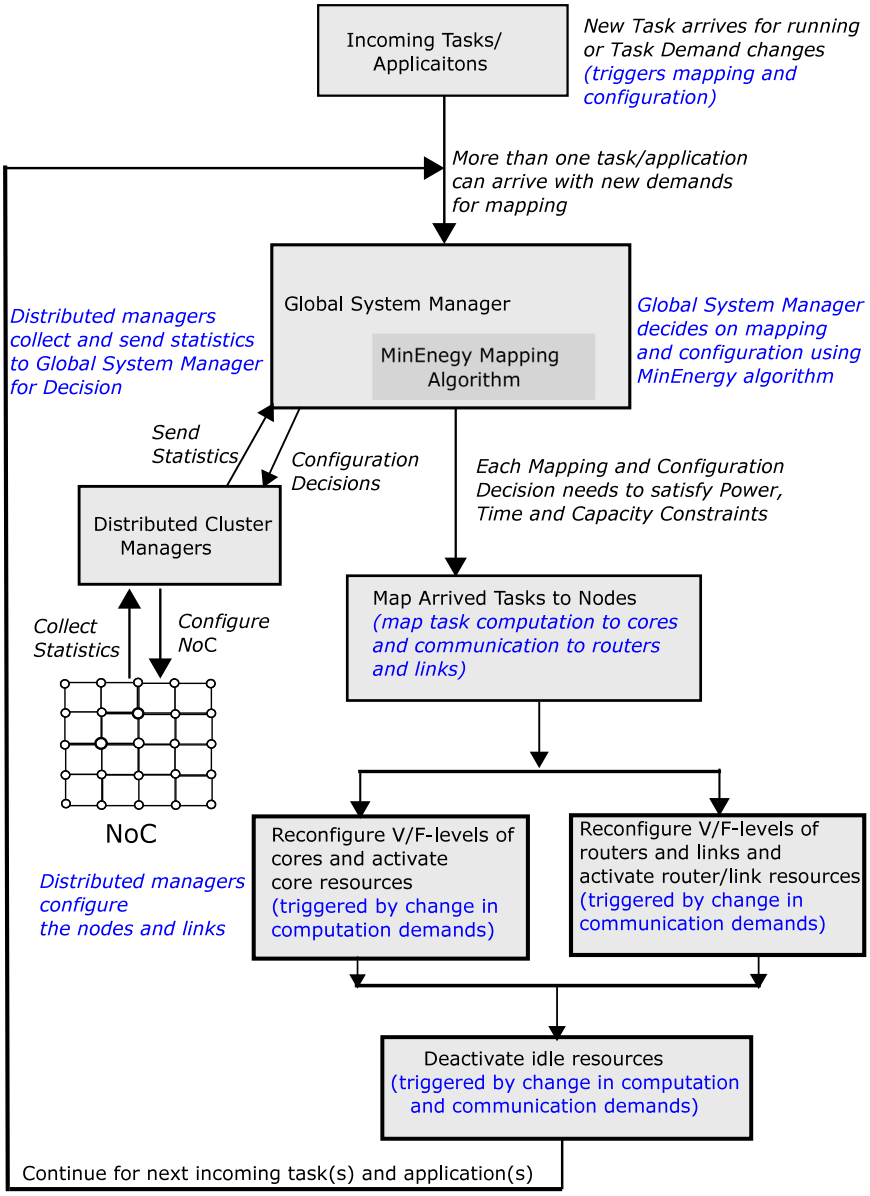


Fig. 4. Proposed Dynamic Mapping and Configuration Solution.

traffic, as detouring increases communication load and it (detouring) can introduce communication bottleneck in the NoC. That is why, XY-routing algorithm is used as a shortest-path routing in 2D-mesh NoC. We are adopting this strategy as we are trying to reduce both overall power consumption and hotspots in our solution. However, our mapping approach works for any kind of NoC topology with minor modification in the communication load calculation. Additionally, the proposed approach is deadlock-free because of the use of XY-routing (dimension ordered routing). Like some dark silicon solutions, we are not deactivating a router when only the corresponding core is not carrying traffic (no task mapped to that core). Instead, we shutdown a router (and connected resources) only when all the links and the core attached to that router are not being utilized.

During run-time, based on the mapping solutions, cluster managers assign only the required voltages to the nodes/links depending on the traffic demand at the nodes/links after the mapping or release of the tasks. Five different voltage-levels and corresponding frequency-levels are used in this work: 0.8 V/1 GHz, 0.9/1.5 GHz, 1.0 V/1.8 GHz, 1.1 V/2 GHz, 1.2 V/2.5 GHz. A cluster manager increases or decreases the voltage levels of the resources depending on the change in loads after the mapping or release of the tasks. Cluster managers assign only the required amount of the voltage levels to the resources depending on the loads. This dynamic assignment of voltages further reduces the power consumption and increases the flexibility of meeting application demands (while meeting power budgets) compared to the static assignment of (maximum) voltages.

6.4. Task migration

For optimal mapping in run-time systems, tasks may need to be migrated (move or swap) among cores. For example, an incoming task may need to run in a big core, while a big core is being utilized by a small task. In this case, it might be optimal to migrate a small task to another little core, and then map the incoming big task to a big core. Task migration can also be performed if that migration reduces overall power consumption and/or energy hotspots. Task migration involves a lot of overhead, e.g., saving variables and route the information through NoC to the migrated core. So, the overhead of task migration needs to be considered justifying its effectiveness in terms of overall energy consumption and penalty of task migration. A feasibility study of

supporting task migration in multi-core system was discussed in [44]. A task migration mechanism for distributed many-core operating systems presented in [45].

7. MinEnergy dynamic mapping algorithm

Our designed LP model in Section 5 achieves optimal mapping of the tasks, but its high computational complexity of our designed LP model restricts its application to run-time mapping. For example, for 50-tasks mapping in 64-core NoC, the total possible mapping solutions are 64_{50}, which is a very large number. To this end, we develop a mapping algorithm with linear time complexity for fast run-time mapping and configuration at run-time. The global system manager with the help of this mapping algorithm map incoming tasks to the nodes in NoC. The greedy algorithm is applied in the selection of both applications/tasks and the processor nodes, as outlined below.

7.1. Task arrival and selection

The task (and application) is selected for mapping on the first-come-first-serve basis. If several tasks (and applications) arrive at once, then the tasks (applications) are sorted in a descending order according to their total computation and communication demands. The demands of an application is presented by: $TD(A_k) = e_{cmp} \cdot \sum_{m=1}^{M} (cmp)_k^m + e_{com} \cdot \sum_{\forall e_k^{mn} \in G} (com)_k^{mn}$, where G is the task graph. Similarly, the demands of a task can be presented by: $TD(T_k^m) = e_{cmp} \cdot (cmp)_k^m + e_{com} \cdot \sum_{\forall e_k^{mn} \in G, n \in G'} (com)_k^{mn}$, where G' is the partial task graph that has been mapped so far. The task (application) with highest demand is mapped first, as it can be mapped with more options for reducing energy and hotspots.

7.2. Node selection and resource configuration

Once a task is selected, we examine all hypotheses of placing the task onto every possible available node (that satisfies all the computation, communication, and power budgets constraints). The node with the lowest overall NoC load is chosen to map the task. In the event of a tie in the lowest overall load, capacity utilization is used as a tie-breaker, where the lowest utilized node is chosen to avoid overloading a node so as diminishing hotspots. First task mapping of an application needs special care as this is the anchor point for that application. The first task is mapped to a node with the highest number of available neighbors around it (to accommodate other incoming tasks of the same application for lower communication loads), and with the lower existing node load (to prevent a node to be overloaded with computation, and communication from other tasks). Therefore, for the first task of an application, we use a node selection factor (nsf), which is the weighted sum of the number of available neighbors (N_i) and the reciprocal of existing computation load (ρ_i) at node i : $nsf_i = \delta_1 \cdot N_i + \delta_2 \cdot \frac{1}{\rho_i}$, where δ_1 and δ_2 are the scaling parameters. The node with the highest selection factor, i.e., $max\{nsf_i | 1 \leq i \leq N\}$ is chosen to map the first task. This minimizes the probability of creating energy hotspots by mapping an application away from the higher loaded region(s). The process repeats until all the incoming tasks are mapped or is aborted if the constraints cannot be satisfied. NoC resources are configured and activated based on the selection of the nodes and demands of the tasks. We also explore the task migration options (considering migration penalty) after a certain interval and migrate a task to a different node to reduce hotspots further without increasing total chip energy. Task migration penalty is measured using communication energy consumption (penalty) for routing a task from a source node to a target node via the control network.

The pseudocode of task-to-node assignment and NoC configuration is shown in Algorithm 2.

Algorithm 2: *MinEnergy* Task-to-Node assignment and NoC configuration Heuristic

```
input  : Arrived Application TGs, NoC Topology Graph
output : Task Mapping, NoC Configuration
if multiple applications arrive in the system then
  | select A_k ← max(TD(A_k));
end
else
  | select the application A_k that arrive in the system;
end
for all tasks in the selected application do
  | if multiple tasks arrive in the system then
  |   | find first task F_T ← max(TD(T_k^{F_t}))|F_t ∈ M);
  | end
  | else
  |   | select the first task that arrive in the system;
  | end
  | select first node F_N ← max(nsf_i|i ∈ N);
  | map(F_T) = F_N; update task list excluding F_T;
  | update topology with F_N node load;
  | for all unmapped tasks do
  |   | find next task N_T ← max(TD(T_k^{N_t}))|N_t ∈ M);
  |   | generate candidate node list L_C ← n_i|i ∈ N
  |   |   that satisfy computation and communication capacity,
  |   |   power budgets and task deadline constraints;
  |   | select next node N_N ← min(sum(Load_i|i ∈ N)) in L_C;
  |   | map(N_T) = N_N; update task list by removing N_T;
  |   | update topology with N_N node load and links weight;
  |   | configure V/F-levels of nodes and links;
  | end
end
Repeat till all the arrived applications and/or tasks are mapped;
return map;
```

7.3. Time complexity

During the mapping, we are only concerned with the incoming task(s) in the queue. The algorithm then explores the active (not dark) node options N (that satisfy all the constraints) for mapping a task. As only a few tasks may arrive in the system at a time, the time complexity of the *MinEnergy* mapping algorithm is only linear order of N, $O(cN)$, where c is a constant. In the worst case, c is equal to the total number of tasks of all the applications to be mapped. Therefore, the time complexity of this mapping algorithm is very much suitable for dynamic mapping scenarios.

7.4. Mapping example

A sample mapping of application-1 and application-2 on a 4×4 2D-mesh NoC using *MinEnergy* is shown in Fig. 5. Both computation and communication demands of the applications are presented in the figure and considered for mapping. Node capacity, link capacity, and power budgets are constrained in this example. Fig. 5(a) and (b) show the computational and communication loads distribution using *MinEnergy* algorithm. Fig. 5(c) shows the node and link wise energy distribution, where Fig. 5(d) shows the total energy per node in the NoC. Total energy at a node is calculated by summing up the energy consumption at the node and the associated links of that node. The goal of *MinEnergy* is to minimize the overall energy consumption while minimizing the possibility of any hotspots on the NoC. Because of the heterogeneous loads at different nodes and links of the NoC, it is very practical to apply different V/F-levels (dim silicon) to different resources to reduce power consumption. We also turn-off (dark silicon) resources to save power consumption. Another thing to note is that we allowed multiple tasks to be mapped to an active node (if the capacity allows) instead of activating another deactivated node. This also addresses the problem that could be encountered if the multi-core architecture has lower

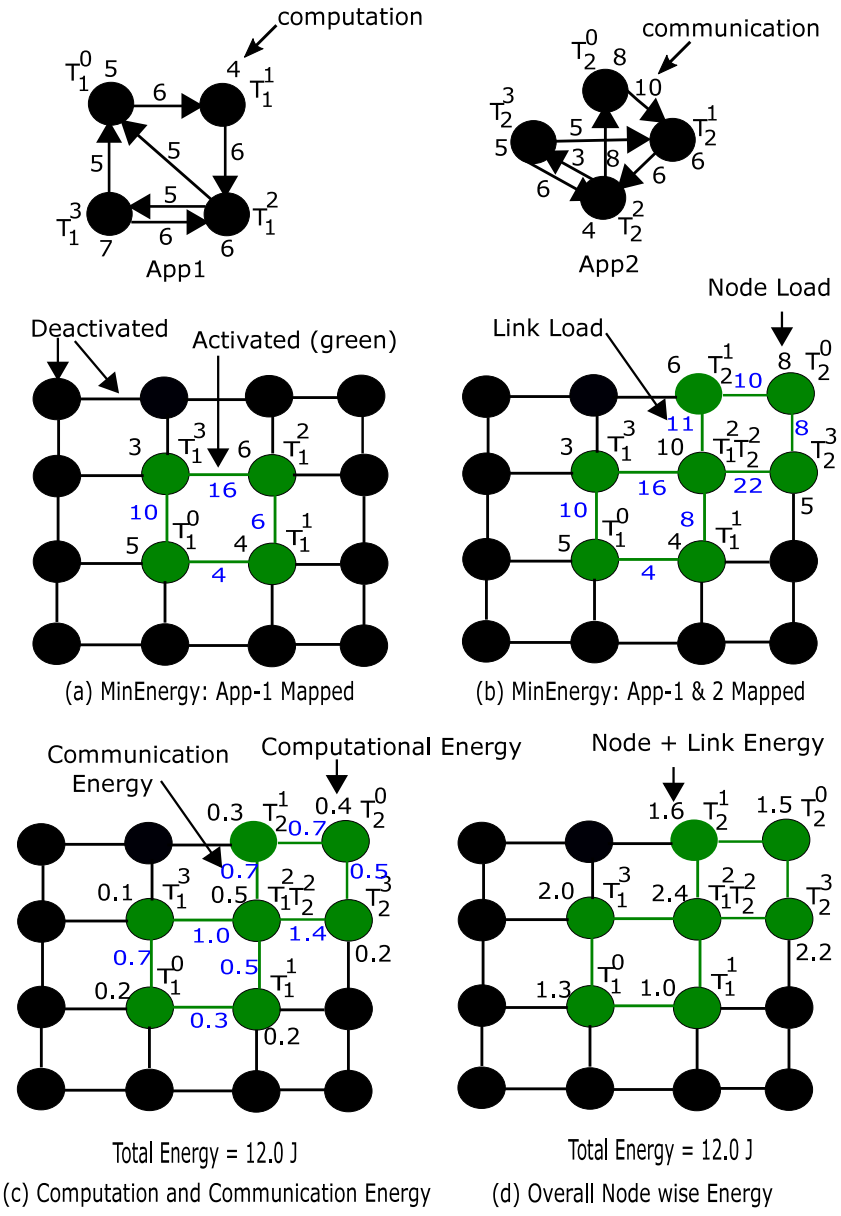


Fig. 5. *MinEnergy* Mapping and Resource Configuration in 4 × 4 NoC.

number of nodes than the number of tasks. After the completion of application-1 mapping, 75% of the total nodes (cores and routers) and 83% of the NoC links are switched-off to save power. Activated nodes and links are shown in green color, where deactivated resources are displayed in black color. After mapping of application-2, percentages of deactivated nodes and links are 56% and 66%, respectively. Nodes and links of the NoC have heterogeneous demands (loads). Therefore, V/F-scaling is applied to both nodes and links of the NoC to reduce power consumption.

8. Simulation & comparison

The IBM CPLEX [46] LP solver is used to implement the LP optimization model described in Section 5. The proposed dynamic mapping approach *MinEnergy* algorithm is implemented using inhouse-developed C++ simulator. Dynamic mapping is simulated by mapping tasks and applications one by one as they arrive in the system. We have implemented a state-of-the-art *Minimum-Path* mapping algorithm by modifying the approach in [8] as the baseline. Task computation demand and allocation, node capacity heterogeneity, multi-application mapping, power-gating, task deadline, chip-level power budget and tile-level power budget constraints are integrated in the *Minimum-Path* algorithm for fair comparison with *MinEnergy* and LP optimizer results. All the reported simulation results for all the mapping and configuration approaches meet the constraints presented in Section 5. Real system experiments are further carried out using gem5 [47] platform with Garnet, a detailed cycle-accurate NoC. The experiments are carried out based on embedded system synthesis benchmark suite (E3S) [48] and random task graphs generated by task graph generator (TGG) [49]. We have simulated our experiments in both small-scale and large-scale networks. For CPLEX, we simulated small-scale NoCs because of large solution space and time complexity.

8.1. Energy evaluation under small-scale NoC

Due to high time complexity, CPLEX cannot produce optimal solutions using the LP model for large NoCs/applications within a finite time. That is why, total, peak energy and node wise energy consumption from all the three approaches (CPLEX, *MinEnergy* and *Minimum-Path*) are compared for small 3 × 3 2D-mesh NoC.

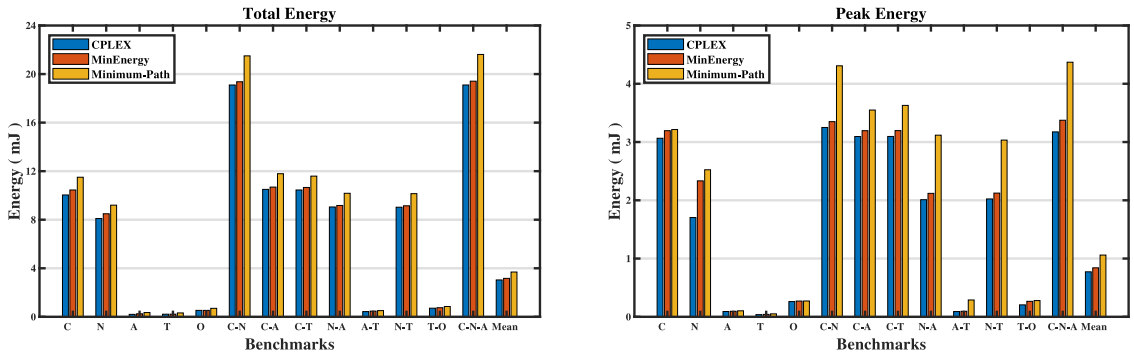


Fig. 6. Energy evaluation under 3 × 3 NoC for E3S Benchmarks.

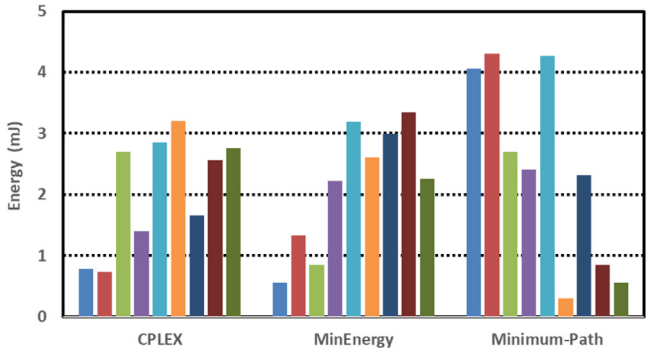


Fig. 7. Node wise energy distribution for C-N Benchmark under 3 × 3 NoC.

As shown in Fig. 6, in 3 × 3 NoC, *MinEnergy* delivers nearly the same total and peak energy as those of optimal solution (CPLEX), where *MinEnergy* provides significantly lower total and peak energy consumption than those of *Minimum-Path* under most of the E3S benchmarks. On average (geometric mean), total energy from *MinEnergy* solution is 4% closer to the optimal solution from CPLEX, and is 16% lower than that of *Minimum-Path*. Similarly, peak energy (hotspot) from *MinEnergy* solution is 8% closer to the optimal solution from CPLEX, and is 26% lower than the solution from *Minimum-Path*. Node wise energy distributions are simulated for mixture of consumer and networking (CN) applications. CN applications have 25 sub-tasks within them. As shown in Fig. 7, *MinEnergy* provides significantly better load-balanced distributions compared to *Minimum-Path*, as several nodes are highly overloaded (and a few nodes are very lightly loaded) in *Minimum-Path* solution. Node wise energy distributions of (*MinEnergy*) are very close to the optimal distributions of CPLEX solution. This shows the near-optimal energy consumption of *MinEnergy* solution and significantly better energy consumption (total energy and hotspots) than that of *Minimum-Path* solution.

8.2. Energy evaluation under large-scale NoCs and applications

As CPLEX cannot produce solution for the proposed LP model (with all the constraints) for more than 9-core NoC in a finite time, we compare the energy, performance and power consumption of *MinEnergy* with only *Minimum-Path* solution for larger NoC. We evaluate the impact of NoC scaling on total energy and hotspots trends for large number of tasks (4 applications with 60 tasks, generated by TGG) and large NoCs (up to 256 cores). As shown in Fig. 8, *MinEnergy* decreases the peak load (hotspot) with the increase in network nodes, as *MinEnergy* allows remote mapping of tasks to reduce power hotspots. But *Minimum-Path* does not decrease the peak load (hotspot) with the increase in NoC size. On average (geometric mean), peak energy consumption from *MinEnergy* is 23% lower than that of *Minimum-Path* for 36-core to 256-core NoCs. For 144-core and 196-core NoCs, peak energy consumption is 40% lower than that of *MinEnergy* solution. *Minimum-Path* only reduces the distance between the communicating tasks and does not use the opportunity of higher available NoC resources in large No. Therefore, *Minimum-Path* is not scalable like *MinEnergy*. Furthermore, total energy from *MinEnergy* is only 1% lower than that of *Minimum-Path*. Because of the non-adjacent mapping consideration of tasks of the same application, total energy consumption slightly increases in *MinEnergy* solution in few cases. So, *MinEnergy* decreases the overall energy or maintain the minimum overall energy consumption while distributing loads more evenly to reduce hotspots.

8.3. Dark silicon impact

We simulate dark silicon impact by making some percentage of cores unavailable for task mapping for both small- and large-scale NoCs and applications. During deactivation of cores, we ensure that the capacity and/or number of cores are enough for running the applications. We also ensure that a few cores are deactivated from each cluster so that dark and active cores distributed uniformly throughout the chip to reduce the thermal impact of cores on each other. We select the cores randomly from each cluster for deactivation. For small 4 × 4 2D-mesh NoC, *MinEnergy*, *Minimum-Path*, and *Random* mapping algorithms are compared for mixture of consumer and networking (CN) applications. *Random* mapping algorithm randomly maps the tasks to available cores (that satisfy the chip constraints).

In all cases, *MinEnergy* has lower peak energy values compared to both *Minimum-Path* and *Random*. For 40% dark cores in 16-core NoC in Fig. 9, *MinEnergy* peaks are 43% and 60% lower than those of *Minimum-Path* and *Random*, respectively, mainly because of balanced load distributions in *MinEnergy*. Routers are not deactivated for this case (16-core NoC). As mentioned before, total load at a node (tile) is calculated by summing up the node (core + router) loads and loads on the associated links (of the corresponding node). That is why, though few cores are deactivated, associated routers (and links) are used for traffic distribution. For 16-core NoC (without dark cores) in Fig. 9, peaks of *MinEnergy* are 92% and 120% lower than those of *Minimum-Path* and *Random*, respectively.

For large-scale NoC and application setting, we simulate with 40% to 80% of dark cores in 12 × 12 2D-mesh NoC for the same 4 applications with 60 tasks as mentioned previously. *MinEnergy* has 15% lower peak and 5% lower total energy compared to those of *Minimum-Path*, as shown in Fig. 10. Because of the lack of scalability property, *Minimum-Path* only performs better when NoC has limited available resources and/or small no. of tasks to be mapped. But the proposed *MinEnergy* performs significantly better for larger NoC and larger no. of tasks/applications.

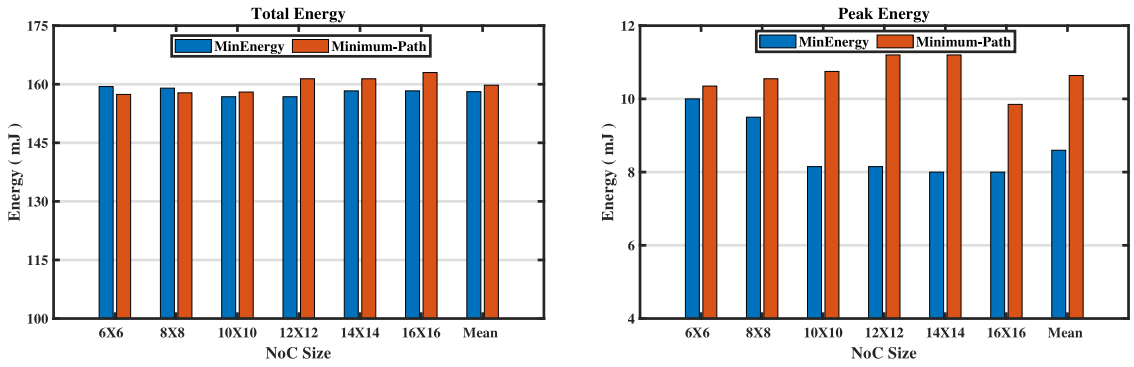


Fig. 8. Energy evaluation under large-network scales for random applications.

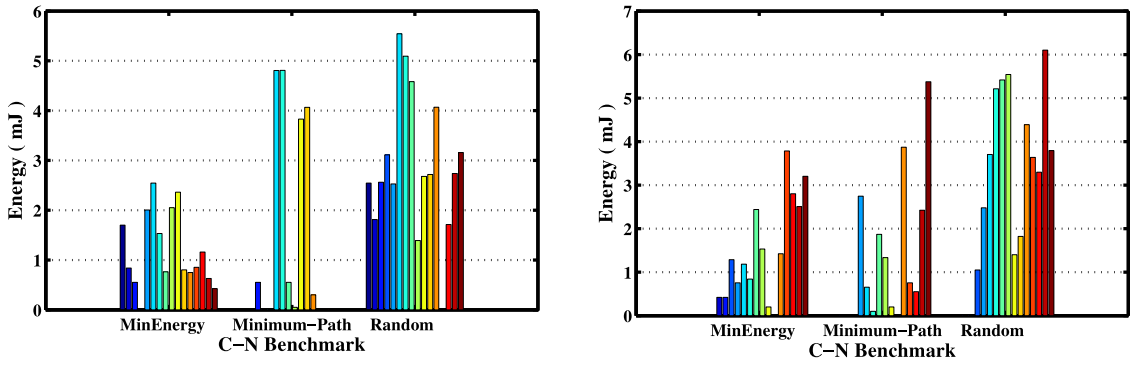


Fig. 9. Energy distribution with and without dark cores under 4 × 4 NoC.

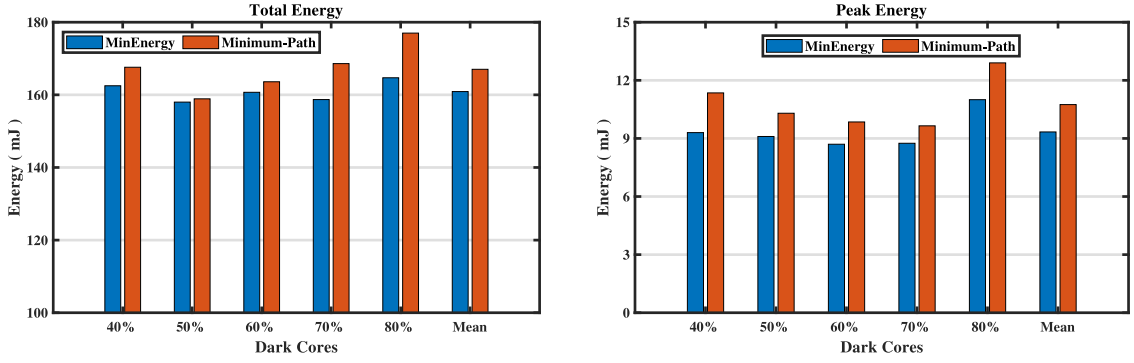


Fig. 10. Dark silicon impact on energy under 12 × 12 NoC.

8.4. Latency and throughput evaluation

Latency and throughput performance of *MinEnergy* and *Minimum-Path* solutions are simulated in gem5 for both small-scale (16-core) and large-scale (64-core) NoCs. We ran the simulations in 16-node and 64-node Alpha architectures connected through 4 × 4 2D-mesh NoC and 8 × 8 2D-mesh NoC, respectively. We ran the simulations for 1,000 cycles and set the maximum number of packets to be injected by each node to 50,000. V/F-level configuration is done at every 100 cycles. Other configuration parameters are kept the same as the default settings in gem5 , such as 3 virtual networks, 4 virtual channels (VC), 4 buffers per VC. gem5 configurations are shown in Table 1.

As seen in Fig. 11, latency and throughput performance of *MinEnergy* exceed *Minimum-Path* by a large margin in most E3S benchmarks. On average (geometric mean), latency and throughput from *MinEnergy* are 27% (lower) and 70% (higher) better than those of minimum-path mapping for all the 13 E3S applications. For example, for *telecom* benchmark ((with 8 sub-applications and 28 tasks)), latency and throughput

Table 1
gem5 Simulation Configuration.

Instruction Set Architecture (ISA)	ALPHA
Interconnection Networks	GARNET
No. of Virtual Networks (VN)	3
No. of Virtual channels (VC) per VN	4
No. of Buffers per VC	4
Network Type	2D Mesh
No. of Cores	16 and 64
No. of Routers	16 and 64
Routing	XY-Routing
Flit Width	128-bit
Max. Packets per Node	50 000
Simulation Cycle	1000

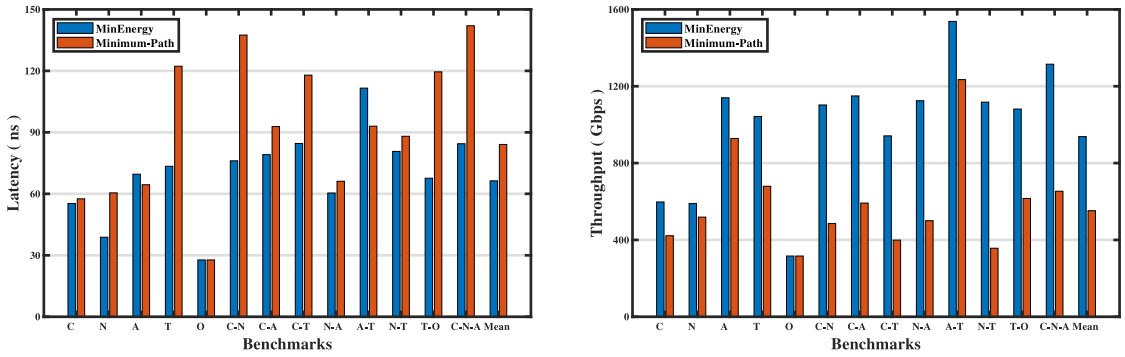


Fig. 11. Latency and throughput evaluation under 4 × 4 NoC for E3S Benchmarks.

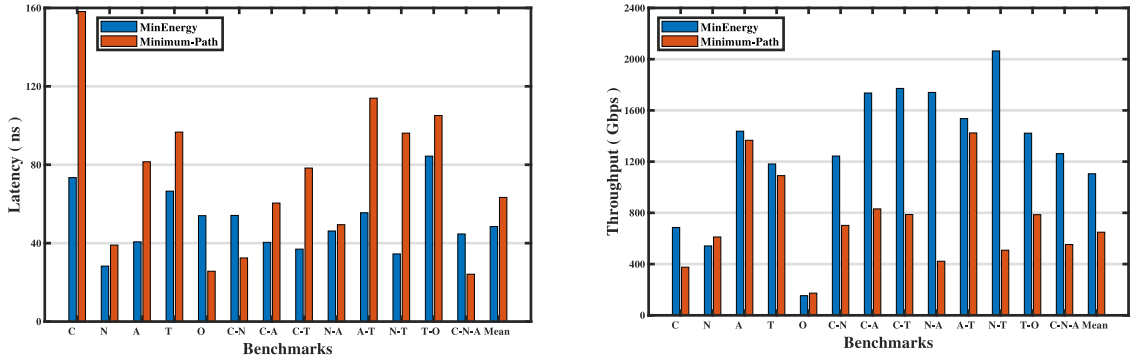


Fig. 12. Latency and throughput evaluation under 8 × 8 NoC for E3S Benchmarks.

from *MinEnergy* are 66% and 34% better than those of minimum-path mapping. Lower total load while reducing hotspots is the main reason for better performance from *MinEnergy*. *Minimum-Path* results in poor performance mainly due to the formation of hotspots, as *Minimum-Path* only considers the communication distance (between tasks) for mapping.

Similarly, for large-scale 64-core NoC, latency and throughput performance of *MinEnergy* are significantly better than those of *Minimum-Path* for the same reasons mentioned above (for 16-core NoC). On average (geometric mean), latency and throughput from *MinEnergy* are 30% (lower) and 70% (higher) better than minimum-path mapping for all the 13 E3S applications, as shown in Fig. 12.

8.5. Power evaluation

As mentioned above, DSENT power tool [50] is used to compare dynamic power consumption of the NoC links and routers. Power consumption of the cores and leakage power of the chip components are not considered in the calculation. DSENT configurations are shown in Table 2. We simulated the dynamic power consumption (dissipation) results for both small-scale (9-core and 16-core) and large-scale (64-core) NoCs. Heterogeneous V/F-level configurations of links and routers from *MinEnergy* approach are evaluated by calculating individual resource power consumption and then summing up all the resources power consumption. We applied heterogeneous V/F-level configurations to CPLEX to get the optimal solution. For fair comparison with *MinEnergy*, we also deactivate the idle NoC resources in the *Minimum-Path* approach.

For comparison with the optimal power consumption, all the three approaches (CPLEX, *Minimum-Path*, and proposed *MinEnergy*) are compared for small-scale 9-core NoC in Fig. 13. On average (geometric mean), power consumption of *MinEnergy* is only 6% higher than that of optimal solution (CPLEX) for all the benchmarks. Power consumption in *MinEnergy* is reduced by 60% compared to *Minimum-Path* for small 3 × 3 2D-mesh NoC. The reasons for reduction of power consumption in *MinEnergy* are dynamic assignment of heterogeneous V/F-levels to the NoC routers and links and balanced distribution of tasks in the NoC. With the heterogeneous voltage/frequency assignment, we can allocate the minimum V/F-levels to meet the requirements of the tasks/applications.

Table 2
DSENT Simulation Configuration.

Technology	22 nm
No. of Virtual Networks (VN)	3
No. of Virtual channels (VC) per VN	4
No. of Buffers per VC	4
Network Type	2D Mesh
No. of Cores	9 and 16 and 64
No. of Routers	9 and 16 and 64
No. of Cores per Router	1
No. of Input Ports	5
No. of Output Ports	5
Flit Width	128-bit
Buffer Model	128-bit
Crossbar Model	Multiplexer crossbar
Switch Allocator Model	MatrixArbiter

As shown in Fig. 14, for small 4 × 4 2D-mesh NoC. *MinEnergy* dynamic mapping consumes significantly lower power than *Minimum-Path* approach under all the E3S benchmarks. On average, power consumption from *MinEnergy* solution is 48% lower than that of *Minimum-Path* because of the more balanced distribution of tasks while satisfying the power budgets constraints. *Minimum-Path* approach tries to serve the application(s) with the minimum NoC resources (imbalanced load distribution in NoC), which results in high hotspots. Because of the balanced distribution of traffic, *MinEnergy* (with the help of distributed resource management technique) can assign the minimum V/F-levels (to the NoC resources), which reduces power consumption significantly (as power is proportional to the product of V- and F-levels).

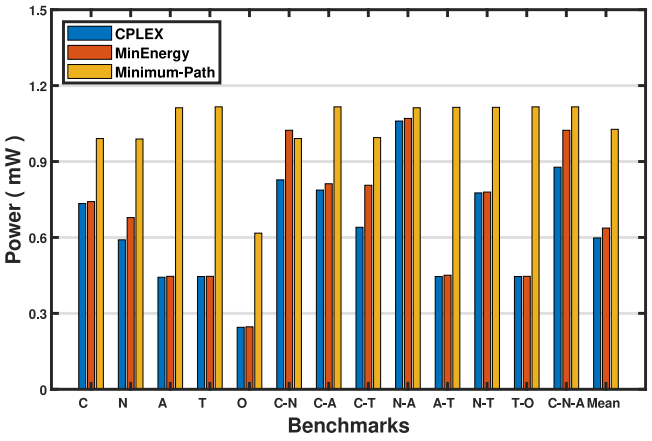


Fig. 13. Power evaluation under 3 × 3 NoC for E3S Benchmarks.

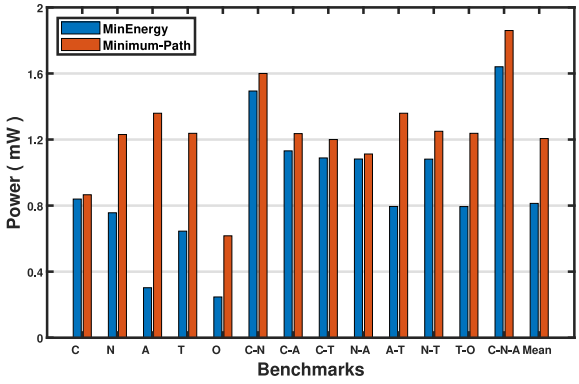


Fig. 14. Power evaluation under 4 × 4 NoC for E3S Benchmarks.

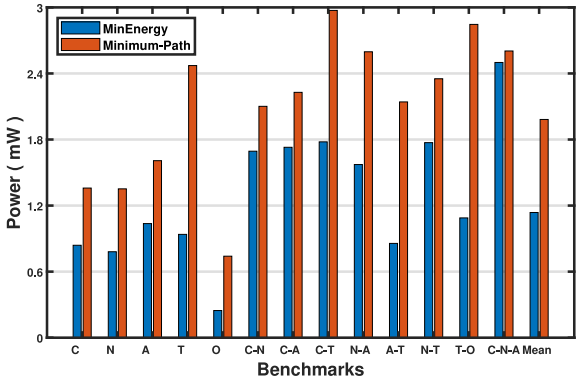


Fig. 15. Power evaluation under 8 × 8 NoC for E3S Benchmarks.

We observe the similar power consumption improvement in *MinEnergy* solution for 64-core NoC, as shown in Fig. 15. On average, power consumption from *MinEnergy* solution is 74% lower than that of *Minimum-Path*. We can see that with the increase in NoCs size (from 9-core to 16-core to 64-core), power improvement increases in *MinEnergy* solution (compared to *Minimum-Path* solution), which shows the scalability property of *MinEnergy*. So *MinEnergy* provides significantly better power consumption than that of *Minimum-Path* while having better NoC latency and throughput performance.

9. Conclusion

The computation and communication capacity, time, and power constrained application mapping and configuration problem in heterogeneous multi-core NoCs has been presented to address the power and thermal challenges in the dark silicon era. The linear programming optimization of task placement and resource configuration has been modeled and implemented. *MinEnergy* mapping heuristic is proposed for fast exploration of mapping and configuration decisions for large NoCs and applications dynamically at run-time systems. Distributed resource management strategy with the help of a global system manager and distributed cluster managers for task mapping and NoC configuration has been proposed for scalable solution in NoC based multi/many-core systems. Dynamic voltage-frequency scaling and power-gating techniques have been applied to the nodes and links of the NoCs to reduce power consumption while meeting the performance requirements. State-of-the-art *Minimum-path* mapping has been implemented for further evaluation of our proposed solution. Simulation results demonstrate that the proposed mapping and configuration framework minimizes overall chip power and hotspots for energy-efficient NoC in multi/many-core systems.

10. Acknowledgement

The authors would like to thank anonymous reviewers and the editor for valuable recommendations to improve this article.

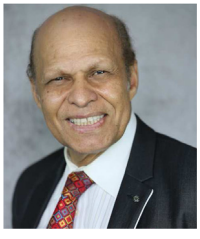
References

[1] M. Kas, Toward on-chip datacenters: A perspective on general trends and on-chip particulars, J. Supercomput. 62 (1) (2012) 214–226.
[2] Cerebras: Wafer Scale Engine, https://www.cerebras.net/product/.
[3] Yosemite, https://engineering.fb.com/core-data/introducing-yosemite-the-first-open-source-modular-chassis-for-high-powered-microservers/.
[4] M.B. Taylor, A landscape of the new dark silicon design regime, IEEE Micro 33 (5) (2013) 8–19.
[5] R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, A.R. LeBlanc, Design of ion-implanted MOSFET's with very small physical dimensions, IEEE J. Solid-State Circuits 9 (5) (1974) 256–268.
[6] M.B. Taylor, Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse, in: Proceedings of ACM/IEEE Design Automation Conference, DAC, 2012, pp. 1131–1136.
[7] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, D. Burger, Dark silicon and the end of multicore scaling, IEEE Micro 32 (3) (2012) 122–134.
[8] S. Murali, G. De Micheli, Bandwidth-constrained mapping of cores onto NoC architectures, in: Proceedings of IEEE Design, Automation and Test in Europe, DATE, 2004, pp. 896–901.
[9] S. Murali, L. Benini, G. De Micheli, Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees, in: Proceedings of ACM/IEEE Asia and South Pacific Design Automation Conference, ASP-DAC, 2005, pp. 27–32.
[10] J. Hu, R. Marculescu, Energy and performance aware mapping for regular noc architectures, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 24 (4) (2005) 551–562.
[11] C. Marcon, A. Borin, A. Susin, L. Carro, F. Wagner, Time and energy efficient mapping of embedded applications onto NoCs, in: Proceedings of ACM/IEEE Asia and South Pacific Design Automation Conference, ASP-DAC, 2005, pp. 33–38.
[12] S. Murali, M. Coenen, A. Radulescu, K. Goossens, G. De Micheli, A methodology for mapping multiple use-cases onto networks on chips, in: Proceedings of IEEE Design, Automation and Test in Europe, DATE, 2006, pp. 1–6.
[13] C.-L. Chou, R. Marculescu, Incremental run-time application mapping for homogeneous NoCs with multiple voltage levels, in: Proceedings of International Conference on Hardware/Software Codesign and System Synthesis, CODES + ISSS, 2007, pp. 161–166.
[14] G. Chen, F. Li, S. Son, M. Kandemir, Application mapping for chip multiprocessors, in: Proceedings of ACM/IEEE Design Automation Conference, DAC, 2008, pp. 620–625.
[15] M.F. Reza, D. Zhao, H. Wu, M. Bayoumi, Hotspot-aware task-resource co-allocation for heterogeneous many-core networks-on-chip, Computers & Electrical Engineering 68 (C) (2018) 581–602.
[16] M.F. Reza, D. Zhao, H. Wu, Task-Resource Co-Allocation for Hotspot Minimization in Heterogeneous Many-Core NoCs, in: Proceedings of ACM International Great Lakes Symposium on VLSI, GLSVLSI, 2016, pp. 137–140.

[17] C.-L. Chou, R. Marculescu, Contention-aware application mapping for Network-on-Chip communication architectures, in: Proceedings of IEEE International Conference on Computer Design, ICCD, 2008, pp. 164–169.
[18] B. Yang, L. Guang, T.C. Xu, A.W. Yin, T. Santti, J. Plosila, Multi-application multi-step mapping method for many-core Network-on-Chips, in: Proc. NORCHIP, 2010, pp. 1–6.
[19] M. Fattah, M. Ramirez, M. Daneshtalab, P. Liljeberg, J. Plosila, CoNA: Dynamic application mapping for congestion reduction in many-core systems, Proceedings of IEEE International Conference on Computer Design, ICCD, 2012, pp. 364–370.
[20] M. Fattah, M. Daneshtalab, P. Liljeberg, J. Plosila, Smart hill climbing for agile dynamic mapping in many-core systems, in: Proceedings of ACM/IEEE Design Automation Conference, DAC, 2013, pp. 1–6.
[21] X.-H. Wang, P. Liu, M. Yang, M. Palesi, M.C. Huang, Energy efficient run-time incremental mapping for 3-D networks-on-chip, J. Comput. Sci. Tech. 28 (1) (2013) 54–71.
[22] C.-L. Chou, R. Marculescu, User-aware dynamic task allocation in Networks-on-Chip, in: Proceedings of IEEE Design, Automation and Test in Europe, DATE, 2008, pp. 1232–1237.
[23] E. Carvalho, N. Calazans, F. Moraes, Heuristics for dynamic task mapping in NoC-based heterogeneous MPSoCs, in: Proceedings of IEEE/IFIP International Workshop on Rapid System Prototyping, RSP, 2007, pp. 34–40.
[24] M.-H. Haghbayan, A. Kanduri, A.-M. Rahmani, P. Liljeberg, A. Jantsch, H. Tenhunen, MapPro: Proactive runtime mapping for dynamic workloads by quantifying ripple effect of applications on networks-on-chip, in: Proceedings of IEEE/ACM International Symposium on Networks-on-Chip, NOCS, ACM, 2015, pp. 26:1–26:8.
[25] A. Kanduri, M. Haghbayan, A. Rahmani, P. Liljeberg, A. Jantsch, H. Tenhunen, Dark silicon aware runtime mapping for many-core systems: A patterning approach, in: Proceedings of IEEE International Conference on Computer Design, ICCD, 2015, pp. 573–580.
[26] A. kanduri, M. haghbayan, A.M. Rahmani, M. Shafique, A. Jantsch, P. Liljeberg, adBoost: Thermal aware performance boosting through dark silicon patterning, IEEE Trans. Comput. 67 (8) (2018) 1062–1077.
[27] H. Khdr, S. Pagani, M. Shafique, J. Henkel, Thermal constrained resource management for mixed ILP-TLP workloads in dark silicon chips, in: Proceedings of ACM/IEEE Design Automation Conference, DAC, 2015, pp. 1–6.
[28] J. Henkel, S. Pagani, H. Khdr, F. Kriebel, S. Rehman, M. Shafique, Towards performance and reliability-efficient computing in the dark silicon era, in: Proceedings of IEEE Design, Automation and Test in Europe, DATE, 2016, pp. 1–6.
[29] S. Murali, M. Coenen, A. Radulescu, K. Goossens, G. De Micheli, Mapping and configuration methods for multi-use-case networks on chips, in: Proceedings of ACM/IEEE Asia and South Pacific Design Automation Conference, ASP-DAC, 2006, 146–151.
[30] M.A.A. Faruque, R. Krist, J. Henkel, ADAM: Run-time agent-based distributed application mapping for on-chip communication, in: Proceedings of ACM/IEEE Design Automation Conference, DAC, 2008, pp. 760–765.
[31] S. Kobbe, L. Bauer, D. Lohmann, W. Schroder-Preikschat, J. Henkel, DistRM: Distributed resource management for on-chip many-core systems, in: Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS, 2011, pp. 119–128.
[32] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-GHz mesh interconnect for a teraflops processor, IEEE Micro 27 (5) (2007) 51–61.
[33] H. Wang, L.-S. Peh, S. Malik, Power-driven design of router microarchitectures in on-chip networks, in: Proceedings of IEEE/ACM International Symposium on Microarchitecture, MICRO, 2003, pp. 105–116.
[34] N. Hardavellas, M. Ferdman, B. Falsafi, A. Ailamaki, Toward dark silicon in servers, IEEE Micro 31 (4) (2011) 6–15.
[35] M.F. Reza, D. Zhao, M. Bayoumi, Dark silicon-power-thermal aware runtime mapping and configuration in heterogeneous many-core NoC, in: Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS, 2017, pp. 1–4.
[36] Intel-Corporation, Intel Xeon processor-measuring processor power. Revision 1.1 in whitepaper, 2011.
[37] S. Pagani, H. Khdr, W. Munawar, J. Chen, M. Shafique, M. Li, J. Henkel, TSP: Thermal safe power - efficient power budgeting for many-core systems in dark silicon, in: Proceedings of International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS, 2014, pp. 10:1–10:10.
[38] M. Shafique, S. Garg, J. Henkel, D. Marculescu, The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives, in: Proceedings of ACM/IEEE Design Automation Conference, DAC, 2014, pp. 185:1–185:6.
[39] D. Zhu, L. Chen, T.M. Pinkston, M. Pedram, TAPP: Temperature-aware application mapping for noc-based many-core processors, in: Proceedings of IEEE Design, Automation and Test in Europe, DATE, 2015, pp. 1241–1244.
[40] R. Ennals, R. Sharp, A. Mycroft, Task partitioning for multi-core network processors, in: Proceedings of the 14th International Conference on Compiler Construction, 2005, pp. 76–90.
[41] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Press and McGraw-Hill Book Company, 1989.
[42] S. Borkar, Exascale computing - A fact or a fiction? in: Keynote At IEEE International Parallel and Distributed Processing Symposium, IPDPS, 2013.
[43] ARM, ARM, Big.LITTLE technology: The future of mobile, 2011, White paper.
[44] S. Bertozzi, A. Acquaviva, D. Bertozzi, A. Poggiali, Supporting task migration in multi-processor systems-on-chip: A feasibility study, in: Proceedings of IEEE Design, Automation and Test in Europe, DATE, European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 2006, pp. 15–20.
[45] S. Holmbacka, M. Fattah, W. Lund, A.-M. Rahmani, S. Lafond, J. Lilius, A task migration mechanism for distributed many-core operating systems, J. Supercomput. 68 (3) (2014) 1141–1162.
[46] IBM CPLEX Optimization Studio, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.
[47] N. Binkert, et al., The Gem5 Simulator, SIGARCH Comput. Archit. News 39 (2) (2011) 1–7.
[48] R. Dick, Embedded system synthesis benchmarks suites (e3s), http://robertdick.org/tools.html.
[49] TGG: Task Graph Generator, http://sourceforge.net/projects/taskgraphgen/.
[50] C. Sun, C.-H.O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, V. Stojanovic, DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling, in: Proceedings of IEEE/ACM International Symposium on Networks-on-Chip, NOCS, 2012, pp. 201–210.

Md Farhadur Reza is an Assistant Professor of Computer Science in the School of Computer Science and Mathematics at University of Central Missouri. Previously he had worked as a Postdoctoral Researcher at Virginia Tech and at George Washington University. He received his Ph.D. and M.Sc. degrees from the Center for Advanced Computer Studies department at University of Louisiana at Lafayette in 2017 and 2014, respectively. He earned his M.B.A degree from the Institute of Business Administration at University of Dhaka in 2011. He attained his B.Sc. degree in Computer Science and Engineering from Bangladesh University of Engineering & Technology in 2005. His research focuses on high performance computing architectures, system-level design for multi-core architectures, system-on-chip, on/off-chip interconnection networks, resource allocation, machine learning, artificial intelligence, etc. He has seven years of professional working experience in telecommunication and software industries.

Danella Zhao is an Associate Professor in the Department of Computer Science at Old Dominion University (ODU). Before joining ODU, she was a Lockheed Martin Endowed Associate Professor in the Center for Advanced Computer Studies at University of Louisiana at Lafayette. She received her M.Sc. and Ph.D. degrees from the Department of Computer Science and Engineering at State University of New York at Buffalo in 2001 and 2004, respectively. She received the NSF Early Career Award in 2009 and JAPS Fellowship Award in 2006. Her research focuses on system-on-chip, network-on-chip, embedded systems, design and test, etc.



Magdy Bayoumi is a Professor and Head of the Department of Electrical & Computer Engineering at University of Louisiana at Lafayette. He received B.Sc. degree in Electrical Engineering from Cairo University, Egypt, M.Sc. degree in Computer Engineering from Washington University, St. Louis, and Ph.D. degree in Electrical Engineering from the University of Windsor, Canada. He has published over 300 papers. He is an IEEE Fellow. He has graduated about 50 Ph.D. and 175 Master's students. His research focuses on VLSI, system-on-chip, internet-of-things, communication and networks, etc.