Power-Thermal Aware Balanced Task-Resource Co-Allocation in Heterogeneous Many CPU-GPU Cores NoC in Dark Silicon Era

Md Farhadur Reza*[1], Dan Zhao†, Magdy Bayoumi‡
Email: farhadur_reza@gwu.edu, zhao@cs.odu.edu, mab0778@louisiana.edu
*Department of Electrical & Computer Engineering, George Washington University, Washington, DC, USA
†Department of Computer Science, Old Dominion University, Norfolk, VA, USA
‡The Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA, USA

***Abstract*—To provide balanced mapping for multiple applications in many-core heterogeneous CPU-GPU systems on network-on-chip (NoC) in dark silicon era, power-thermal aware task-resource co-allocation with reconfigurable NoC framework is proposed in this work. Task allocation and resource configuration problem is initially formulated using linear programming (LP) optimization model to search for an optimal solution. Distributed resource management in heterogeneous CPU-GPU systems with a fast balanced mapping heuristic is proposed to minimize power and thermal hotspots and improve performance at run-time while meeting the computation, communication, power and thermal budgets constraints of many-core NoC in dark silicon. We have also implemented a state-of-the-art *minimum-path* contiguous mapping for comparisons. Simulations under real-world benchmarks and platforms show that the proposed dynamic load-balanced mapping strategy improves NoC latency and throughput by 50-100% (while providing near-optimal solution) compared to *minimum-path*.**

I. Introduction

With the advancement and miniaturization of transistor technology, hundreds of cores can be integrated on a single chip. A single chip with heterogeneous cores (CPU/GPU) can be designed to serve the heterogeneous demands of the various customers and applications. Industries and academicians are working to integrate many-cores on a single chip that can be comparable with the big data-center solution, for example, Kilo-Core Chip [2]. Many-core chip solution can reduce companies' operations and maintenance costs tremendously in most areas, e.g., power consumption, cooling, and infrastructure costs for current and future challenges in traditional x86-based data-centers [14]. For communication in many-core systems, NoC offers several important benefits over the traditional bus in terms of scalability, parallelism, throughput, and energy efficiency [7]. For example, a 9-core chip is illustrated in Fig. 1, where heterogeneous CPU-GPU tiles are placed in a 3×3 2D-mesh NoC. Each CPU/GPU tile contains a processor, its local cache, a slice of the shared last-level cache, and a router for data/control transmission between the tiles via varying bandwidth links. The heterogeneous (big/little) routers are interconnected to form a NoC. Memory controller (MC) is used for memory-traffic transfer to/from off-chip memory.

The rise of dark silicon raises several design challenges in many-core systems. 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 [21]. For example, transistor density increases by $2x$, where transistor power efficiency improves by $1.4x$ in every processor technology, which raises $2x$ deficit in power budget [21] following the same trend of core integration trend. Because of this power deficit, 21% and 50% of the chip resources may have to be dark (deactivated) at $22nm$ and $8nm$ process technology, respectively [9]. To address power limitation, dark silicon (power-gated) and dim silicon (voltage/frequency scaled) become the major solutions for current power limitation in many-core chip [12], [21]. Relative power consumption of NoC has been increasing with the increase in the number of cores in a chip, e.g., NoC consumes 36% of total chip power in 16-tile MIT Raw [22]. As a significant power source,

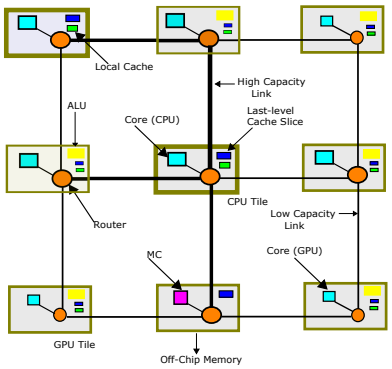


Fig. 1: Heterogeneous link and core configuration in 3×3 NoC for heterogeneous CPU-GPU systems

NoC resources need to be configured dynamically to decrease pressure on power budget (by deactivating or using least capacity to meet traffic demands). Besides power limitation, a component of the chip can not exceed a manufactured thermal limitation at any instant of time as high temperature can burn the chip. Thermal hotspots increases several failures (e.g., electromigration) and can cause permanent damage to the chip. Following are the design challenges that have been addressed in this paper for many-core NoC: power budget, thermal hotspot, dark silicon, resource heterogeneity, and run-time resource configuration.

Wide range of applications with unpredictable demands can enter many-core architectures for running, which makes on-the-fly mapping and configuration a challenging job. Efficient dynamic resource allocation and activation technique is needed to meet power and thermal budgets of many-core chip while satisfying demands of the applications. We assume het-

[1]Work was partially completed at University of Louisiana at Lafayette

erogeneous CPU and GPU cores are spread out in many-core NoC. Load-balancing and resource configuration are needed in heterogeneous CPU-GPU cores NoC to minimize power and thermal hotspots. The challenges are to configure the bandwidth of the links and voltage-levels of the nodes during run-time as applications may not be known in advance, and scenario of an application may even change at run-time.

The contribution of this work is outlined below:

- *CPU-GPU Resource Management:* Distributed resource management strategy in heterogeneous CPU-GPU systems using CPUs for system management and latency-sensitive tasks and GPUs for throughput-intensive tasks has been proposed. The global manager takes the mapping decision (select cluster and cores) for global optimal solution, and cluster managers configure the NoC resources (e.g., bandwidth / voltage assignment, activation, deactivation).
- *Dynamic Task-Resource Co-Allocation in heterogeneous CPU-GPU Cores NoC:* Task-resource mapping considering power and thermal budgets in heterogeneous CPU-GPU systems are dynamically co-designed for energy orchestration. This work configures the NoC link widths and node voltages depending on the applications traffic, for lowering the waste of energy compared to the homogeneous link width and node voltage-level solution.
- *Balanced Mapping:* Balanced task mapping to handle computation and communication workload variations of multiple applications is a crucial issue in chip design. By supporting load-balanced distribution and dark/dim silicon features, this work reduces power and thermal hotspots and improves latency/throughput performance.
- *LP Optimization Model:* Task mapping and resource configuration problem to minimize energy hotspots while meeting the computation, communication, power, and thermal constraints of many-core NoC in dark silicon is formulated using LP for optimal solution.

Related work are discussed in Sec. II. LP formulation of this work is presented in Sec. III. We discuss our distributed resource management approach and load-balanced mapping algorithm in Sec. IV. Simulations are demonstrated in Sec. V.

II. Related Work

The challenges of application-architecture mapping are to select the number of nodes and links from the pool of heterogeneous resources while meeting the chip budgets and applications demands. Most of the previous task-resource allocation work [6], [10], [11], [13], [15] only focus on communication distance between nodes during mapping. Though minimum-path algorithms reduce overall communication load, they create load-imbalance and contention because of uneven load distribution in different NoC regions. More importantly, only few work [17]–[19] have addressed the power and thermal limitations during task-resource co-allocation while considering both computation and communication demands of an application. In this work, we consider computation and communication loads of the tasks jointly to reduce both computational and communication power and thermal hotspots in many-core system. Furthermore, this is the first work (to the best of our knowledge) that addresses the presence of heterogeneous CPU and GPU cores in a chip for efficient allocation of multiple applications as per the demands of applications. For fair comparisons with our solution, we have modified a state-of-the-art minimum-path algorithm in [15] by integrating computation, communication, power, thermal, and deadline constraints.

III. LP Problem Formulation

The goal of the LP optimization is to develop the formulation for allocating tasks and configuring heterogeneous CPU-GPU cores NoC (link width, node voltage-level, and power-gating resources) while balancing load distribution (and power/thermal dissipation) in the chip. An application A_k can have M variable tasks, and m^{th} task of A_k is represented by T_k^m, as presented in Fig. 2. Computation demand of T_k^m is $(cmp)_k^m$. Two tasks m and n of an application A_k can communicate with each other with communication requirement $(com)_k^{mn}$. Let $(\alpha_k^m)^i$ be a binary variable to

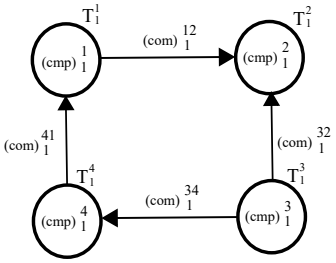


Fig. 2: An example of application task graph G

indicate if a task T_k^m is mapped to node i, i.e.,

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

$(\alpha_k^m)^i$ is to be determined in order to achieve the real-time dynamic mapping and configuration optimization goal. Next, we outline the chip and task constraints, and the optimization objective.

A. Chip-Level Power Budget Constraint

The LP optimizer needs to give an optimal solution without exceeding the designed power budget pw_{budget} of the chip. Overall power consumption is calculated by summing up the power consumption of the chip components (CPU-GPU core, router, and link). Dynamic power consumption is calculated by multiplying the frequency with the square of the voltage-level of components. Besides dynamic power consumption, resource leakage power and activation/deactivation power also contribute to the overall chip power consumption.

$$(\sum_{1\leq i,j\leq n} pw_{ij}^{link} + \sum_{1\leq i\leq n}(pw_i^{router} + pw_i^{core}) + pw_{activation}^{resource} + pw_{deactivation}^{resource} + pw_{leakage}^{resource}) \leq pw_{budget}, \forall 1 \leq i,j \leq n. \quad (2)$$

B. CPU/GPU Node-Level Thermal Constraint

Thermal heat dissipation th_i at a CPU/GPU tile (node) is calculated by summing up the temperature dissipation of the router and associated core and links attached to that

tile. Temperature of a node i is measured using ambient temperature (t_a), power consumption (pw_i), surface area (sa), and thermal resistance (tr).

$$th_i = t_a + (pw_i \cdot tr_i)/sa_i, \forall 1 \leq i \leq n. \tag{3}$$

Thermal dissipation of a node i can not exceed maximum temperature TH_{max} to avoid exceeding thermal safe power (TSP) of a chip. TSP is configurable [16] based on the application demands, NoC size, and chip power consumption.

$$th_i = (\sum_{1 \leq i,j \leq n} (th_{ij}^{link} + th_i^{router} + th_i^{core} + th_{activation}^{i,ij} + th_{deactivation}^{i,ij} + th_{leakage}^{i,ij})) \leq TH_{max}, \forall 1 \leq i,j \leq n. \tag{4}$$

C. Communication and Link Configuration Constraints

Different NoC links carry heterogeneous traffic among resources (e.g., CPU to GPU, CPU to CPU, GPU to GPU). At run-time, performance requirements of an application may change or unknown application with new demands can arrive at the system. Hence, there is a need for NoC link capacity reconfiguration during run-time. A communication from node p to node q go through a set of links depending on the underlying routing algorithm. Several communication flows can share a link l_{ij}. $Path_{pq}$ contains all the links needed for communication between a source and a destination. Total communication traffic on a link l_{ij} can not exceed the maximum communication capacity BW_{ij} on that link.

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

The bandwidth BW_{ij} of NoC links are heterogeneous and configurable as applications requirements are usually heterogeneous. BW_{ij} is calculated by multiplying the link width configuration with the corresponding link frequency f_{link} as,

$$BW_{ij} = (\sum_{h=1}^{H} lw_{ij}^h \cdot B_h) \cdot f_{link}, 1 \leq i,j \leq n. \tag{6}$$

lw_{ij}^h is a binary variable to indicate the configured h^{th} link width configuration B_h (e.g., 32/64/128 bits) for l_{ij}.

D. CPU/GPU Computation and Voltage-Level Constraints

Heterogeneous CPU and GPU nodes have different computational capacity. The sum of the computation demand of all the tasks assigned to CPU/GPU node i can not exceed computational capacity P_i^{cap} of node i, i.e.,

$$\rho_i = \sum_{\substack{1 \leq k \leq K \\ 1 \leq m \leq M}} (cmp)_k^m \cdot (\alpha_k^m)^i \leq P_i^{cap}, \forall 1 \leq i \leq n. \tag{7}$$

The voltage-level of a node i is configured based on the mapped computation load in (7).

E. 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:

$$\sum_{\substack{1 \leq p \leq n \\ 1 \leq m \leq M \\ 1 \leq k \leq K}} ((st_{cmp})_k^m)^p \cdot (\alpha_k^m)^p + \sum_{\substack{1 \leq p \leq n \\ 1 \leq m \leq M \\ 1 \leq k \leq K}} ((T_{cmp})_k^m)^p \cdot (\alpha_k^m)^p + \sum_{\substack{1 \leq p,q \leq n \\ 1 \leq m,n \leq M \\ 1 \leq k \leq K \\ \exists l_{mn} \in E}} ((T_{com})_k^{mn})^{pq} \cdot (\alpha_k^m)^p \cdot (\alpha_k^n)^q \leq D_k^m. \tag{8}$$

F. Objective Function

Our objective is to jointly minimize the peak computation and communication energy (power and thermal) dissipation of all the nodes and links in the chip, i.e.,

$$Minimize : max(e_{cmp} \cdot \rho_i \cdot P_i^{ON} + e_{com} \cdot \sum_{\forall j \in l_{ij}} bw_{ij} \cdot L_{ij}^{ON}), \quad 1 \leq i,j \leq n, \tag{9}$$

subject to the constraints given in (2), (4), (5), (7), and (8). P_i^{ON} and L_{ij}^{ON} are binary variables to indicate active node i and link ij, respectively. Total energy dissipation at a node i is calculated by summing up its computation energy and all the communication energy on the associated links. In $22nm$ technology, computation e_{cmp} and communication e_{com} energy coefficients are 45 *picojoule (pJ)* per floating point operations (we assume 8-bit per computational operation for simplicity), and 65 *pJ* per communicating bit, respectively [5].

IV. Balanced Mapping and Resource Management

BalancedMap distributed resource management strategies and mapping algorithm are presented in this section.

A. Distributed Resource Management in Many CPU-GPU Cores NoC

We choose distributed resource management (cluster managers and a global manager) over fully centralized mechanism (only a global manager) because of its (distributed) scalability in many-core NoC. A chip is assumed to be partitioned into a number of clusters. Every cluster can have a different no. of CPUs and GPUs and a cluster should have at least one CPU node for cluster resource management, as shown in Fig. 3. The clustering decision is another NP-hard problem, and that (clustering) is not the focus of this paper. A cluster manager (CM) monitors and configures the resources within that cluster. CMs (CPUs) transfer the resource status of the cluster to the global manager (GM) via control network in NoC. A CPU with shortest distance from all the CMs is assigned as GM. Both CM and GM also do the computation of the task as other cores. CMs collect following statistics in fixed intervals for the corresponding cluster: task-to-core mapping status, resources activation/deactivation status, capacity utilization of resources, and tasks start-finish time. 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. GM selects the cluster and CPU-GPU cores for mapping the tasks using *BalancedMap* mapping algorithm (which will be discussed soon) and clusters resource status to achieve near-optimal solution. GM classifies task as throughput and latency-intensive by seeing task demands and real-time (hard or soft) deadline constraint. GM maps the computation intensive (and highly parallel) tasks to GPUs and latency-sensitive tasks to CPUs. Distributed CMs configure the NoC resources based on the mapping decision.

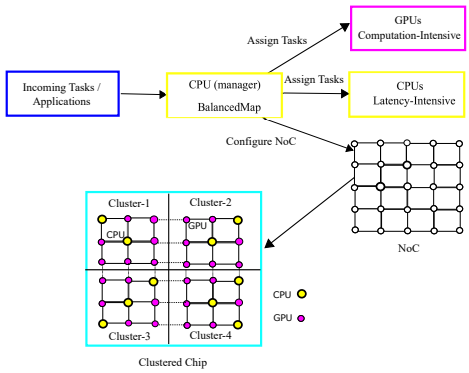


Fig. 3: Mapping and configuration in a CPU-GPU system

The solution we adopt to minimize overall energy is to bring down the total dynamic and leakage energy. We achieve this minimum overall energy by shutting down idle NoC resources, and by reducing link bandwidths and node voltage-levels at run-time based on the task demands. In this work, power-gating technique is used to shutdown a resource (node/link) if there is no task using that resource. When a new task/application arrives, we try not to activate power-gated resources if already active resources can meet the demands of the new task/application. After the mapping of all the tasks in the queue, CMs deactivate the resource that has been idle for a certain time. We can set the threshold time for deactivating resources by studying applications nature. Shutdown feature can save a significant amount of power in terms of leakage power when many resources are not carrying any traffic because of the lower demand of the application(s). Wake-up signal is used to send the signal to activate a resource. CMs assign only the required voltages to the CPU/GPU cores and required bandwidths to the links. Five different voltage-levels are used in this work: 0.8, 0.9, 1.0, 1.1, and 1.2 Volt. CMs increase or decrease the voltage-levels and link bandwidths of the resources depending on the change in application traffic. This dynamic configuration of voltages and bandwidths further reduces the power and thermal consumption.

B. BalancedMap Mapping Algorithm

The greedy load-balanced algorithm (shown in Algorithm 1) is applied in the selection of both applications/tasks and the processor nodes, as outlined below.

1) Task/Application Selection and Task Classification: The task and application are selected based on the arrival time and/or demands of the tasks and applications. Arrival time is used to serve the task and application on first-come-first-serve basis. Total demand of a task is calculated by summing up the computation demand and communication demands with other tasks. If multiple tasks arrive at the same time, then the task with the highest total demand is mapped first, as it can be mapped with more options for reducing power/thermal hotspots and mapped more contiguously (with communicating tasks) for lowering communication energy and delay. Similar strategy applies for selecting an application when multiple applications arrive at the same time.

Task is classified as throughput-intensive and latency-sensitive by using task demands, deadline, and predefined computation and communication thresholds. This task classification facilitates proper node (CPU/GPU) selection in the node selection stage.

2) CPU and GPU Node Selection: All the available GPU and CPU node options are explored for mapping throughput-intensive and latency-sensitive tasks, respectively. The node with the lowest peak energy is chosen to map the selected task. Some node options are not considered for mapping for not meeting the LP constraints. Node capacity utilization is used as a tie-breaker if more than one node options provide the best solution. Lowest utilized node is selected to

input : Arrived Application TGs, NoC Topology Graph
output : Task Mapping, NoC Configuration
if *multiple applications arrive in the system* **then**
  sort $A = \{A_k | k \in K\}$ in $L_A \leftarrow TD(A_k)$;
end
else
  select the application 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 $FT \leftarrow max(TD(T_k^{F_t}) | F_t \in M)$;
  end
  else
    select the unmapped-task that arrive in the system;
  end
  classify task as computation or communication intensive;
  select first CPU or GPU node $FN \leftarrow max(f_i | i \in N)$;
  map(FT)=FN; update task list excluding FT;
  update topology with FN node load;
  for *all unmapped tasks* **do**
    find next task $NT \leftarrow max(TD(T_k^{N_t}) | N_t \in M)$;
    classify task as computation or communication intensive;
    generate candidate node list $L_C \leftarrow n_i | i \in N$ with link bandwidth configuration along routing paths that satisfy capacity, power and thermal budgets and task deadline constraints;
    select next CPU or GPU node $NN \leftarrow min(max(Load_i | i \in N))$ in L_C;
    map(NT)=NN; update task list by removing NT;
    update topology with NN node load and links weight;
    configure node voltage and links width;
  end
end
Repeat till all the arrived applications and/or tasks are mapped;
return map;

Algorithm 1: *BalancedMap* task-to-node assignment and NoC configuration heuristic

avoid overloading nodes for reducing hotspots. This balanced distribution also reduces thermal-effects of active nodes (to each other) and increases the thermal safe power of the chip.

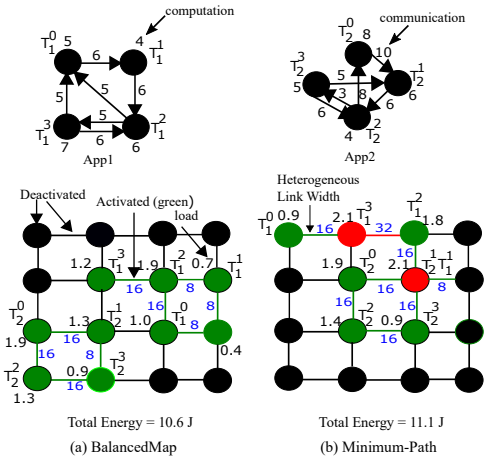


Fig. 4: *BalancedMap* mapping and configuration in 4×4 NoC

3) NoC Resource Configuration: Depending on the selection of the CPU/GPU nodes, NoC link widths and node voltage-levels are configured. Link width is configured by activating required number of wires on a link. Node voltage-level is set based on the required task computation at a node. Traffic-balancing on links and nodes are performed to avoid overloading CPU/GPU nodes and associated links. A link is deactivated if that link is not carrying any communication traffic. We shut-down a router only when all links and core attached to that router are not being utilized. A core, which is not a manager, is deactivated if no task is mapped to it.

4) Task Migration: Our mapping procedure performs task migration after a fixed interval to further minimize hotspots in NoC. Because of the transistor aging and variability degradation, different nodes with the same capacity can consume different energy for running a same task. At run-time, node utilization can exceed its warning threshold. So we explore the options for migrating a task to a different CPU/GPU node and perform the migration if that migration reduces energy hotspots considering migration penalty and NoC constraints. Task migration penalty is measured in communication time and energy consumption for migrating and routing a task from a source node to a target node via NoC control network. Cluster managers control the task migration by sending control packets to the nodes.

5) Mapping Comparison and Time Complexity: We compare *BalancedMap* (BM) and *Minimum-Path* (MP) mappings for two random applications on a 4×4 2D-mesh NoC. As we see in Fig. 4, BM reduces the hotspots by distributing the loads evenly throughout the network. MP introduces hotspots (red nodes or links) even though huge percentage of NoC resources are unused (black). Furthermore, BM decreases the total chip energy consumption compared to MP. Though BM has slightly higher number of active (green in Fig. 4) NoC node and link resources, it (BM) meets the power and thermal budgets constraints. MP needs more NoC link resources for meeting application demands compared to BM as seen from high link widths in the MP mapping solutions. We also found

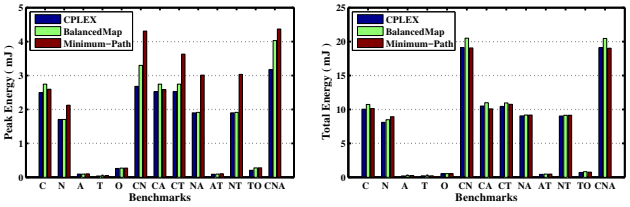


Fig. 5: Energy evaluation under 3×3 NoC for E3S Benchmarks

that MP needs higher power and thermal budgets to provide a contiguous mapping solution, while BM can produce a load-balanced mapping solution within limited chip budgets.

The time complexity of BM heuristic is $O(KMN)$, where K, M, and N are the number of applications, tasks, and NoC nodes, respectively. As only few number of applications/tasks may arrive in the system at a time, time complexity reduces to linear order of $O(CN)$, where C is a constant. For example, for two applications with 25 tasks, CPLEX and BM take 3.9 hours and 20 milliseconds, respectively, for an optimal solution. Therefore, BM algorithm is suitable for fast mapping and configuration in run-time systems.

V. Simulation & Comparison

Task-resource co-allocation LP optimization model in Sec. III is implemented by commercial IBM CPLEX [1] LP solver. *Minimum-Path* (MP) and *BalancedMap* (BM) mapping heuristics are developed using inhouse-developed C++ simulator. Heterogeneous CPU-GPU systems are emulated by integrating cores of different computational capacity. Mapping solutions and NoC configurations from the heuristics are sent to gem5 [4] platform with Garnet [3], a detailed cycle-accurate interconnection network model, to evaluate the NoC performance in terms of latency and throughput. Power consumption is evaluated using DSENT [20]. Embedded system synthesis benchmark suite (E3S) [8] is used for experiments as this work is applicable to embedded systems. E3S consists of consumer (C), auto-industry (A), networking (N), telecom (T) and office-automation (O) applications.

Peak, total, and node-wise chip energy of all the above mentioned approaches for E3S benchmarks are compared. 3×3 is used for CPLEX because of the large search/design space of optimal solution. As shown in Fig. 5, in 3×3 NoC, peak energy from BM is within 0-10% of the optimal solution, where BM provides up to 60% lower peak energy (hotspot) than the MP solutions. Total energy consumption of BM solutions are also very close to the optimal and MP solutions. For some benchmarks, total energy from BM is slightly higher (around 5%) than MP, as BM distributes loads to remote nodes because of higher resource availability (lower utilization) compared to task demands. BM distributes loads uniformly like optimal solution (CPLEX), while MP creates energy hotspots, as shown in Fig. 6.

We simulate dark silicon impact by making 10-70% of cores (CPU/GPU) unavailable for task mapping in Fig. 7. Dark and active cores are spread out on the NoC (instead of separate regions for dark cores) to evenly distribute power and temperature density. BM and MP mappings are compared in 64-core dark NoC for telecom application, which has 28

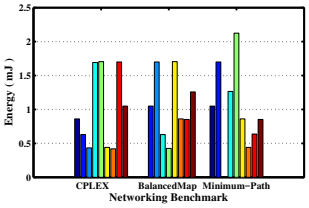


Fig. 6: Node wise Energy Distribution under 3×3 NoC

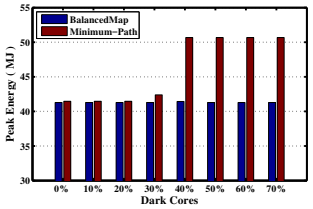
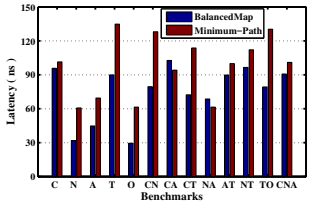


Fig. 7: Dark Silicon impact on Peak Energy under 8×8 NoC



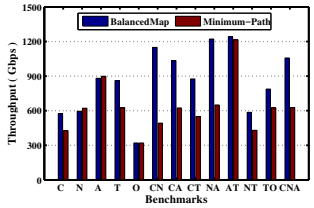


Fig. 8: Latency/throughput evaluation under 4×4 NoC for E3S Benchmarks

TABLE I: gem5 configuration

Instruction Set Architecture (ISA)	ALPHA
CPU Frequency	2GHz
Interconnect Frequency	1GHz
Interconnection Networks	GARNET
No. of Virtual Networks(VN)	3
No. of Virtual channels(VC) per VN	4
No. of Buffers per VC	4
Network Size	4×4 Mesh
No. of Cores	16
No. of Routers	16
Routing	XY-Routing
Flit Width	128-bit
Max. Packets per Node	50000
Simulation Cycle	1000

TABLE II: DSENT configuration

Technology	22nm
Operating Frequency	1 GHz
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
No. of Routers	9
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
Switch Allocator Model	MatrixArbiter

tasks (including subtasks). As seen in Fig. 7, peak energy from MP solution has increased significantly after 40% dark cores. On the other hand, BM provides a good solution (by not increasing hotspots) with the limited active resources.

Latency and throughput performance of BM and MP solutions are simulated in gem5. Heterogeneous link bandwidths in gem5 are modeled by modifying link latencies (to mimic bandwidths) in on-chip Garnet network. gem5 configurations are shown in Table. I. As seen in Fig. 8 for most of the E3S benchmarks, latency and throughput from BM are 50-100% lower and 40-100% higher than MP solutions, respectively. Load-balanced distribution (considering heterogeneous node capacity) in heterogeneous NoC is the main reason for better latency and throughput performance from BM.

For power consumption evaluation of mappings, DSENT configurations are shown in Table II. Power consumption of heterogeneous NoC link bandwidth architecture is evaluated by calculating individual link power consumption and then summing up all the links power consumption. Because of the uniform distribution of loads, higher number of NoC resources are activated in load-balanced solutions (BM and CPLEX) compared to MP solutions for some E3S benchmarks. This results in higher power consumption in load-balanced solutions than MP in some cases, as shown in Fig. 9.

VI. Conclusion

Balanced task-resource co-allocation in many-core heterogeneous CPU-GPU systems in reconfigurable dark NoC under the computation and communication capacity, power, thermal, and deadline constraints has been addressed in this paper. The LP optimization of task placement and

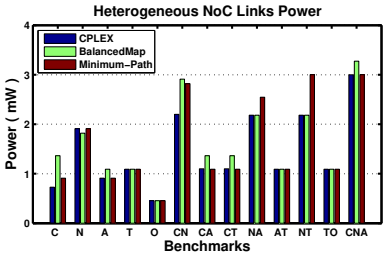


Fig. 9: Power evaluation under 3×3 NoC for E3S Benchmarks

resource configuration has been modeled and implemented. Distributed resource management strategy and load-balanced mapping heuristic are proposed for fast exploration of mapping and configuration solution space for large dark NoC and applications.

References

[1] IBM CPLEX Optimization Studio, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[2] World's First 1,000-Processor Chip, https://www.ucdavis.edu/news/worlds-first-1000-processor-chip/.

[3] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *Proc. ISPASS*, pages 33–42, 2009.

[4] N. Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, 2011.

[5] S. Borkar. Exascale computing - a fact or a fiction? In *Keynote at IPDPS*, 2013.

[6] C.-L. Chou and R. Marculescu. Incremental run-time application mapping for homogeneous nocs with multiple voltage levels. In *Proc. CODES+ISSS*, pages 161–166, 2007.

[7] G. De Micheli et al. Networks on chips: From research to products. In *Proc. DAC*, pages 300–305, 2010.

[8] R. Dick. Embedded system synthesis benchmarks suites (e3s), http://robertdick.org/tools.html.

[9] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. *IEEE Micro*, 32(3):122–134, 2012.

[10] M.-H. H. et. al. MapPro: Proactive runtime mapping for dynamic workloads by quantifying ripple effect of applications on networks-on-chip. In *Proc. NOCS*, pages 26:1–26:8, 2015.

[11] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila. Smart hill climbing for agile dynamic mapping in many-core systems. In *Proc. DAC*, pages 1–6, 2013.

[12] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, 2011.

[13] J. Hu and R. Marculescu. Energy and performance aware mapping for regular noc architectures. *IEEE Trans. TCAD*, 24(4):551–562, 2005.

[14] M. Kas. Toward on-chip datacenters: A perspective on general trends and on-chip particulars. *J. Supercomput.*, 62(1):214–226, 2012.

[15] S. Murali and G. De Micheli. Bandwidth-constrained mapping of cores onto noc architectures. In *Proc. DATE*, pages 896–901, 2004.

[16] S. Pagani et al. TSP: Thermal safe power: Efficient power budgeting for many-core systems in dark silicon. In *Proc. CODES*, pages 1–10, 2014.

[17] M. F. Reza, D. Zhao, and M. Bayoumi. Dark silicon-power-thermal aware runtime mapping and configuration in heterogeneous many-core NoC. In *Proc. ISCAS*, pages 1–4, 2017.

[18] M. F. Reza, D. Zhao, and H. Wu. Task-resource co-allocation for hotspot minimization in heterogeneous many-core nocs. In *Proc. GLSVLSI*, pages 137–140, 2016.

[19] M. F. Reza, D. Zhao, H. Wu, and M. Bayoumi. Hotspot-aware task-resource co-allocation for heterogeneous many-core networks-on-chip. *Computers & Electrical Engineering*, 68(C):581–602, 2018.

[20] C. Sun et al. DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *Proc. NOCS*, pages 201–210, 2012.

[21] M. B. Taylor. A landscape of the new dark silicon design regime. *IEEE Micro*, 33(5):8–19, 2013.

[22] M. B. Taylor et al. The raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, 2002.