Leakage Power Aware Task Assignment Algorithms for Multicore Platforms

Gayathri Ananthanarayanan
School of Information Technology
Indian Institute of Technology Delhi
New Delhi, India
gayathri@cse.iitd.ac.in

Smruti R. Sarangi and M. Balakrishnan
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi, India
{*srsarangi,mbala*}*@cse.iitd.ac.in*

***Abstract*—Increased power density and high temperatures are looming issues in many-core processors. Technology scaling trends, cooling limitations and stringent application requirements make these issues rather difficult to handle. Consequently, it is imperative to design solutions that are effective and also scale well with increasing core counts. In this work, we present an application mapping framework `LeakOpt`, which aims to minimize the total power consumption of manycore processors. We first demonstrate the implications of lateral heat conduction on leakage power consumption and show that heat-spread aware task assignment can significantly impact the total power consumption. We formulate the mapping problem as an optimization problem and design a family of algorithms to solve it heuristically. We present simulation results that shows reduction upto 27.12% in leakage power consumption relative to worst case task mapping for a variety of workloads. Heuristic based mapping schemes perform 2600x faster (for 225 cores) while still within 2.5% of best case results. We further evaluate the same algorithms on a real hardware (`TILE-Gx36`™) and show that these techniques can reduce leakage by upto 18.22% on average. Results on hardware are consistent with the simulation results as far as the relative effectiveness of various heuristics is concerned.**

***Keywords*-leakage power; lateral heat conduction; task mapping; temperature; multicore**

I. Introduction

With the end of Dennard scaling, supply voltage can no longer be decreased with every new generation of technology scaling leading to substantial increase in onchip power density. Increasing power densities result in high on-die temperatures. High on-chip temperature increases the chip power consumption due to positive temperature leakage power feedback loop. This also has detrimental effect on ageing and reliability of the processors [1]. Many-core processors are becoming severely power constrained to the extent that it is predicted that only a fraction of cores can be turned on because of the capacity to remove heat. This is known as *DarkSilicon* [2, 3] in literature. It is therefore imperative that we design efficient solutions to control the chip power consumption.

In this work, we study the problem of lateral heat conduction across cores in multicore processors and examine various task assignment algorithms to mitigate its harmful effects such as increased leakage power. Lateral heat conduction is defined as the heat flow between different cores on a die. Traditionally, the EDA research community has not considered this problem to be very important because most of the heat is dissipated via the integrated heat spreader and heat sink. The lateral thermal resistance across silicon is roughly four times that of the vertical thermal resistance [4]. However, the effect is not insignificant in today's processors considering the high on-chip power densities. The work by Henkel et al. [3] shows the new trends in dark silicon from a thermal perspective and also the need for spatio-temporal mapping decisions. Therefore, for processors with large number of cores, spatial heat interference (lateral heat conduction) is becoming more relevant.

To illustrate the implications of lateral heat conduction on the leakage power consumption, we simulate a 64 core tile processor in Ansys Icepak [5] and study the heat transfer across cores for various task to core mappings. Figures [1a, 1b and 1c] shows the resultant chip temperature for various mappings. The linkage between the lateral heat conduction and leakage power consumption results in increasing core temperature. This gets amplified due to the cyclic dependence between temperature and leakage power. Thus, these simulations reveal the fact that the leakage power sensitive task to core assignment can significantly alleviate hotspot formation and also reduce the total power consumption. We observe similar results on a real processor as well (refer figures 2a and 2b).

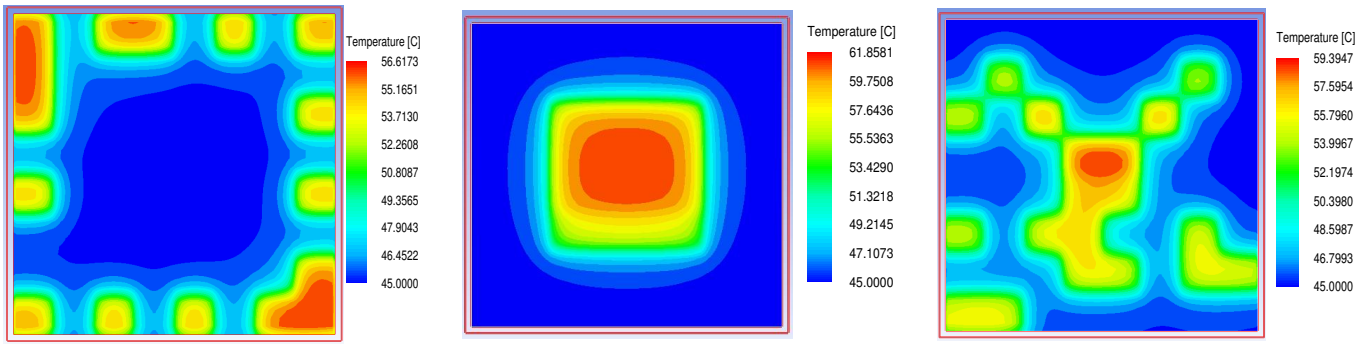
In summary, this paper is organized as follows. We first demonstrate the implications of lateral heat conduction on chip power consumption. We introduce the required background and related preliminaries in Section II and then formulate power aware task mapping problem as an optimization problem in Section III. We observe that solutions based on integer programming, or based on maximizing different forms of distance metrics have high computational complexity (typically $O(N^2)$ for placing each task) and thus cannot be used in an online setting. Subsequently, we devise and examine a family of algorithms to solve the mapping problem heuristically in Section IV. We propose approximately $O(log(N))$ time algorithms that underperform `QCQP` by atmost 2.5%, and are orders of magnitude faster. In Section V, we describe our simulation environment and evaluation setup. In Section VI, we first show that our heuristics can significantly reduce power consumption through simulations and validate the same on a 36-core tile processor. Finally we conclude in Section VII.

II. Background and Related Work

In this section, we briefly discuss the background information on the underlying architecture and related power, performance and thermal models.

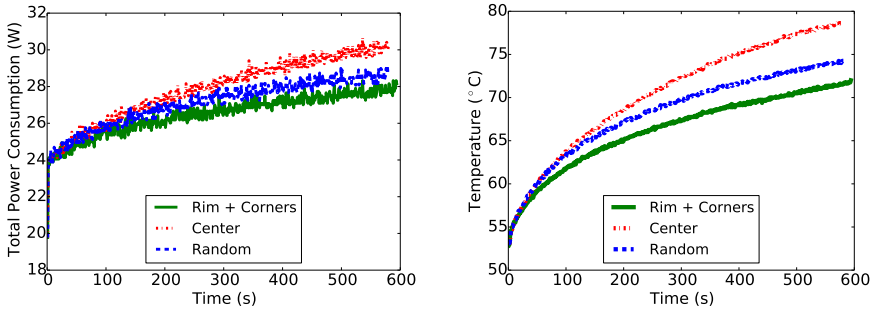
*1) **Architecture Model**:* We consider a tiled processor architecture consisting of N micro-architecturally homogeneous tiles. Each tile comprises a processor core, L1/L2 cache subsystem together with a memory management unit. The tiles in the processor are connected to each other through 2D mesh topology and are organized as a grid of X*Y where $X * Y = N$.

*2) **Workload Model**:* We consider both synthetic and real workloads. During any decision epoch, we assume M ($M \leq N$) independent tasks will be mapped and executed on the processor. For evaluating on real workloads, we construct various multi-program workload mixes by combining various benchmarks from



(a) Rim and Corners (b) Center (c) Random

Figure 1: ANSYS Simulation Thermal map for 16 tasks with different mappings



(a) Chip Power Consumption (b) Chip Temperature

Figure 2: Nine instances of Hmmer workload executed with different mappings on TILE-Gx36™

SPEC 2006 benchmark suite.

3) ***Power Model****:* Each task has two sources of power consumption. The dynamic power due to switching activity is given by $\alpha CV^2 f$ and the leakage power is given by equation[1]. BSIM4 and more recently BSIM-CMG models the dependence of leakage power on temperature and the same is given by the following equation:

$$P_{leakage} \propto v_T^2 * e^{\frac{V_{GS}-V_{th}-V_{off}}{\eta * v_T}} (1 - e^{\frac{-V_{DS}}{v_T}}) \quad (1)$$

v_T is the thermal voltage (kT/q), V_{th} is the threshold voltage, V_{off} is the offset voltage in subthreshold region and η is a constant. The threshold voltage V_{th} decreases with increase in temperature and v_T increases linearly with increase in temperature.

In this work, we consider that each task's steady state average dynamic power consumption profile is known apriori. The power consumption of $\mathbf{task_i}$ executing on $\mathbf{core_j}$ is given by

$$P_{tot_{ij}} = \alpha_i C_i V_{dd}^2 F_j + P_{leakage_j} \quad (2)$$

4) ***Thermal Model****:* `LightSim` [6] uses Hankel transforms to model the heat spread function of a point heat source. It characterizes the lateral heat conduction as the sum of radially symmetric rapidly decaying function and a constant κ. It captures the effect that a primary power source (*due to a task running on a core*) increases the temperature in the neighborhood, and each point in the neighborhood (*adjoining cores*) starts acting as a secondary power source dissipating leakage power. It then uses green's function approach to compute the resultant temperature field while taking into account the leakage temperature feedback loop. We use this temperature estimation framework in our work to calcuate the resultant core temperature.

5) ***Performance****:* The core, L1 and L2 are tied together within a tile in our architecture and runs at a single frequency. We do not consider per-tile DVFS in this work and assume that all the tiles operate at the same frequency. As the tiles are architecturally homogeneous and run at the same frequency, we can say that, the throughput of the task will be the same when executed on any tile.

A. Related work

Task mapping and scheduling is a well researched area and there is a rich literature on task mapping and scheduling solutions [7] targeting power and thermal management. Most of task mapping schemes, suffer from poor scalability issues. Finding an optimal task to core mapping is a NP-hard problem, and thus the associated complexity and overhead grows considerably for large number of cores.

Coskun et al. [8] solve the problem of task allocation in 8 core MPSOCs for various objectives like minimizing and balancing energy, reducing thermal hotspots and temperature gradients using ILP techniques. *Pro-Temp* by Murali et al. [9] uses convex optimization to pro-actively control the temperature of the cores and minimize the power consumption. The work by Ebi et al. [10] presented a hierarchical agent based approach to map tasks to cores with the aim of only minimizing the peak temperature. More recent work of Hu et al. [11] introduced a framework that can efficiently allocate power to each core while considering thermal constraints.

Most of these solutions do not take into account the spatial heat conduction and positive temperature leakage power feedback loop effects. We utilize the insights from the leakage converged

heat spread function characterization (refer Section II-4) to devise heuristics for solving the power aware task mapping optimization problem. We design three algorithms that try to match the optimal solution in Section IV. They primarily use the concept of the region of influence(ROI) of an active core to estimate the temperature spread. We also compare our algorithms to a greedy solution that tries to maximize the distance to all other active cores.

III. Problem Formulation

A. *Optimal Mapping*

Let the M tasks be $\tau_1 \dots \tau_M$ and their dynamic power consumption be $P^d_{\tau_1} \dots P^d_{\tau_m}$. For each of the N cores indexed by i, let P^d_i, P^s_i, P^{tot}_i and T_i be the dynamic, leakage, total power and temperature respectively.

Let X_{ij} denote the mapping between core i and task j.

$$X_{ij} = \begin{cases} 1, & \text{If task } j \text{ is mapped to core } i \\ 0, & \text{otherwise} \end{cases}$$

*1) **Objective Function**:* We aim to minimize the total chip power consumption:

$$\text{Minimize} \quad \sum_{i=1}^{N} P^{tot}_i \tag{3}$$

We also obtain the maximum chip power consumption for a given set of tasks by negating the objective in our optimization formulation. This is done primarily to quantify the maximum impact of task assignment.

*2) **Constraints**:* We have the following constraints on X_{ij}.

- Each task j has to be mapped to a core i and at most one task j can be mapped to a core.

$$\forall j, \sum_{i=1}^{N} X_{ij} = 1, \qquad \forall i, \sum_{j=1}^{M} X_{ij} \leq 1, \quad X_{ij} \in \{0,1\} \tag{4}$$

- The dynamic power for core i is given by:

$$P^d_i = \sum_{j=1}^{M} X_{ij} P_{\tau_j} \tag{5}$$

Leakage power's exponential dependence on temperature can be approximated with piecewise linear models [12]. For the chip operating temperature range of 40°C to 90°C , we assume a linear model of leakage power. The leakage power and total power for core i is given by

$$P^s_i = \alpha + \beta T_i, \quad P^{tot}_i = P^d_i + P^s_i \tag{6}$$

where, α and β values depend on the operating temperature range.

Power Gating: We consider power gating in our model. Any core that is not dissipating any dynamic power is turned off such that the leakage power dissipation is zero. As we have assumed M tasks where $M \leq N$, it is assumed that the rest of the $N - M$ cores have zero dynamic power dissipation. In this case, the modified equation for leakage power is:

$$P^s_i = \alpha + \beta T_i \times \left(\sum_j X_{ij} \right) \tag{7}$$

- The steady state temperature is related to total power consumption through a linear relationship as given by

$$T = AP \tag{8}$$

A_{ik} is the thermal resistance co-efficient between core i and k. Here, A is a symmetric $n \times n$ matrix, T(temperature) and P(power) are $n \times 1$ vectors.

- We set the `chip thermal limit as` T_{max} and this is the maximum allowable temperature that can guarantee safe operation of the chip.

IV. Heuristics

A. *Generic Structure*

Table I: Glossary of terms

Symbol	Definition
τ_i	task i
P_{τ_i}	dynamic power of τ_i
$pointQueue$	Contains all the cores that are unmapped
flp	$flp[i]$ gives the x-y co-ordinates of the i^{th} core on the die
$algoType$	Type of algorithm: {`coolMap, spreadMap, hybrid, greedy` }
$selCore$	Assigned core for task i

We define four algorithms in this section. All of them have a common structure as shown by Algorithm 1. We show a glossary of terms in Table I. The procedure `computeMap` takes as input the type of the algorithm, the dynamic power of the task, a priority queue of cores ($pointQueue$) that are unmapped and a floorplan(flp). For every chip, we pre-characterize the heat spread function (f_{sp}) and store the heat influence matrix $f_{\triangle}Mat$ as proposed in [6].

ALGORITHM 1: `scheduleTasks()`

Input: $algoType$, i, P_{τ_i}, $pointQueue$, flp
Output: $selCore$

```
1  taskQueue ← all tasks to be scheduled ;
2  taskQueue ← sort taskQueue in decreasing order
     of dynamic power;
3  while taskQueue not empty do
4      curTask ← poll head of taskQueue ;
5      P_τi ← Dynamic Power of curTask ;
6      selCore ← computeMap(algoType, i, P_τi, pointQueue, flp);
7  end
```

Procedure computeMap(algoType, i, P_{τ_i}, pointQueue, flp)

```
1  minDist ← ∞ ; minPoint ← −1 ;
2  return A corner core if size(pointQueue) = n;
3  forall the p ∈ pointQueue (in order of descending
     priority) do
4      dist ← calculateCost(p, pointQueue, flp, algoType) ;
5      if dist < minDist then
6          minDist ← dist ;
7          minPoint ← p ;
8      end
9      if algoType ∈ {CoolMap, SpreadMap} then
10         break ;
11     end
12 end
13 deque p from pointQueue ;
14 updateState(algoType) ;
15 return p
```

In Line 3, the procedure `computeMap` loops through all the cores in the $pointQueue$ in descending order of priority. For each core, it computes an algorithm specific cost function. The aim is

to minimize the cost (Lines 5 to 8). For two of our algorithms, `coolMap` and `spreadMap`, the top of the queue is optimal. Hence, there is no need to iterate further and we can break. Lastly, we dequeue the core p, and update the state in Line 14. `computeMap` returns the mapped core, p.

*1) **`spreadMap`** Algorithm:* We observe that not all the cores spread heat equally to neighboring cores (refer Figures 1a, 1b). Especially, cores at the corners and the sides dissipate a lot of heat to the surroundings. Consequently, from the point of view of total power, we should prefer cores at the corners and sides before placing tasks at the center. For each core, we consider the integral of f_Δ, multiplied by -1 as its priority. The integral of f_Δ represents the total increase in chip leakage power if the core is supplied with 1 Watt of dynamic power. We use the framework in [6] to obtain the matrix $f_\Delta Mat$. Each row in $f_\Delta Mat$ specifies the increase in the temperature field, if the i^{th} core is heated by 1 watt. Cores at the corner will have a higher priority and cores at the center will have a lower priority. In this case, $pointQueue$ can be a priority queue implemented as a heap.

*2) **`hybrid`** Algorithm:* There is a shortcoming with the `spreadMap` algorithm. It places too many tasks side by side. It might be preferable to move some tasks towards the center if there are already a lot of tasks at the rims of the die. Hence, we modified the `spreadMap` algorithm as follows. Let us consider a threshold ν for the integral of the spread function. If the integral is less than the threshold, then its priority is the same as that computed by the `spreadMap` algorithm. However, if it is more than ν, then we fall back to the `greedy` scheme. The priority is equal to the mean square distance to all the mapped cores plus a large constant, Ψ. We use Ψ to ensure that the algorithm will fall back to the greedy mode of operation only when it has exhausted all the cores that have an integral less than ν. Here, again the $pointQueue$ can also be an array based heap. It is possible to traverse it in linear time and also extract the minimum in $O(log(n))$ time.

*3) **`coolMap`** Algorithm:* In this approach, we maintain a dynamic estimate of the current temperature of each core. We map each task to the coldest available core. In this case, the `updateState` function is more elaborate. Now, we make the assumption that leakage power is more or less linearly dependent with temperature; hence, the principle of superposition applies. Therefore, as we map or unmap a core, we can add or subtract its temperature field from the temperature map for all the cores in the die. The temperature field is captured by f_Δ and we ignore κ because it is the same for all cores. If we heat a core by 1 Watt, then the augmented temperature field is obtained by scaling f_Δ with the applied dynamic power, and then adding the resultant field to the existing temperature map. An efficient data structure for saving this function is a matrix. We need not update all the cores in the matrix since f_Δ is a rapidly decaying function. We need to update all the cores in the region of influence of f_Δ, which is small.

*4) **`greedy`** Algorithm:* Each task is mapped to an idle core which maximizes the mean square distance to all the mapped cores on the die. The aim here is to minimize the effect of lateral heat conduction. The `updateState` function will be empty. The `calculateCost` function will simply calculate the mean square distance between the core p and all the cores that have already been mapped to tasks. In this case $pointQueue$ can be a simple linked list.

We summarize all the algorithms in Table II.

Table II: Summary of the algorithms

Algorithm	Cost	Time Complexity
`greedy`	mean square distance to all the mapped cores	$O(N^2)$
`spreadMap`	-1 * (integral of f_Δ)	$O(log(N))$
`hybrid`	`spreadMap` + `greedy`	$O(N^2)$ (for a fraction of the cores)
`coolMap`	-1 * (estimated temperature)	$O(log(N)+k)$ (k is the number of elements in f_Δ)

V. Evaluation

In this section, we first describe our experimental setup, simulation environment and validation hardware. We then evaluate the efficiency of our algorithms in comparison.

A. *Experimental Setup*

To evaluate the efficiency of our algorithms, we performed experiments over a wide range of floorplans from 20 tiles (5x4) to 225 tiles (15x15). We assume the die area to be constant (400 mm^2) as per the 2011 ITRS report [13]. Furthermore, we consider 20 core processor at 32nm to be the baseline. Table III summarizes configuration of the baseline processor.

For SPEC2006 workloads, we obtain the power traces and CPI stacks of the benchmarks for the baseline chip configuration (refer Table III) using Sniper 6.0 architectural simulator integrated with McPAT [14]. We use the scaling methodologies described in [2, 3, 13] to quantify the technology scaling effects and obtain the dynamic, leakage power, frequency and threshold values for other technology nodes accordingly. We compute the leakage temperature trends for various technology nodes using MASTAR 5.0.51 [13].

Table III: Baseline Processor Configuration

Parameter	Value	Parameter	Value
Cores	20	Technology	32 nm
Frequency	2.33 GHz	V_{dd}	1.0V

Table IV gives the scaling projections for various technolgy nodes. The values in the columns of Frequency, V_{dd} and Power in Table IV are normalized values (normalized with 32nm as the base) while the other columns represent absolute values.

Table IV: Scaling Projections

Tech (nm)	No.of cores	Frequency	V_{dd}	Power	V_{th} (V)
32	20	1.00	1.00	1.00	0.285
22	32	1.31	0.95	0.73	0.220
16	64	1.69	0.93	0.55	0.175
11	121	2.16	0.90	0.41	0.179
8	225	2.72	0.90	0.31	0.186

We use the thermal estimation framework of [6] and compute the leakage converged Green's functions (center, edge, corner) for a chip with the parameters shown in Table V.

Table V: Thermal estimation framework configuration

Parameter	Value	Conductivity	Value
Die size	400 mm^2	Silicon	130 W/m-K
Spreader thickness	3.5 mm	Spreader	370 W/m-K
Heatsink thickness	24.9 mm	Heatsink	237 W/m-K

B. Simulation Methodology

We have carried out a set of simulation experiments in which we consider a bag-of-tasks workload model. We assume that different cores in the processor run independent tasks. We evalaute our algorithms over a wide range of floorplans from 20 cores to 225 cores and with a set of multi-program workloads constructed from combining benchmarks from the SPEC 2006 benchmark suite. We also consider 3 utilization scenarios for evaluation, with 25%, 50% and 75% of the cores active and running tasks. We replicate each workload by 2x and 3x for the 50% and 75% active core scenarios, respectively. We compare the four algorithms – `greedy`, `spreadMap`, `coolMap`, `hybrid` – to the optimal solutions computed by Quadratic Constrained Quadratic Program (`QCQP`) and two other schemes namely `checkered` and `pinned`. For computing the results of `QCQP`, we use the R interface of Gurobi Mixed Integer Programming solver [15]. The simulations were performed on an Intel Xeon CPU E5-2640 v3 server operating at 2.60 GHz, and having 20MB of L3 cache.

C. Hardware Validation Methodology

We evaluate the proposed heuristics and quantify their benefits on a real hardware platform. We use 36-core `TILE-Gx36™` processor as our hardware in this work. Each of the 36 processor cores is a complete 64-bit processor with 32KB L1 and 256KB L2 caches and running at a clock frequency of 1.2GHz. The power consumption of the core is measured using the `INA219_0-VRM` power monitor. The power monitor measures the power by sensing the voltage and current by voltage drop over a shunt resistance.

We query the power monitors using the `tile board test kit(BTK)` interface. We sample the core power every 500ms and record the power and timestamps. For each workload run, we record start and end timestamps and average all the power samples collected during this interval. We validate the proposed heuristics on 20 different workload combinations for 3 different utilization scenarios as mentioned in Section V-B. `TILE-Gx36™` processor architecture does not have a hardware floating point unit. We therefore use only SPEC 2006 Integer workloads for validation. We first compute the task to tile mapping for each of the proposed heuristics. We then use `sched_setaffinity()` to set the affinity of the task to the desired tile.

VI. Results

To understand the potential impact of minimal assignment, we first perform a limit study to find the difference between the best and worst solutions for leakage power. We then compare our heuristics with the best solution to estimate the efficiency of the proposed heuristics. The maximum leakage power is obtained with the worst mapping. The % reduction in leakage power consumption(`LPC`) achieved using the proposed heuristics is obtained in comparison with this maximum by computing $((\mathbf{1} - \frac{\mathbf{LPC_{heuristic}}}{\mathbf{LPC_{worst}}}) * \mathbf{100})$.

We also include two other schemes namely the `checkered` [16] and `pinned` in our comparison. In the checkered algorithm, each task is mapped to an idle core such that active cores and inactive cores are interleaved in a checkerboard pattern. The pinned algorithm maps the task to the first available idle core in an indexed linear circular list.

A. Simulation Results

Figures [3a, 3b and 3c] shows the % reduction obtained while running workload mix consisting of `lbm, gobmk, sphinx3, omnetpp and mcf` benchmarks for `25%, 50%` and `75%` utilization scenarios respectively. For 20 different multi-programmed workload combinations, we present the average % savings obtained in figures[4a, 4b and 4c].

We observe that for both 25% and 50% activity, upto 36 cores the gains are limited ($<= \mathbf{7}\%$). But for larger number of cores, there is approximately a $\mathbf{20}\%$ difference ($\mathbf{24.7}\%$ and $\mathbf{21.6}\%$ for 225 cores). The difference mainly arises by assigning tasks in a way such that there are many cores out of the range of influence of active cores. However, when we increase the percentage of utilization, it becomes hard to find many such cores, and thus optimal assigment does not result in sizeable benefits. For 75% utilization, the difference drops to $\mathbf{13.8}\%$.

We also observe that all the heat spread based algorithms (except for greedy algorithm) are within 2% of the optimal solution. For 225 cores, `coolMap` takes only $\mathbf{0.146}$ seconds whereas `QCQP` solution takes around $\mathbf{6.5}$ minutes for computing the complete assignment(refer Table VI).

Table VI: Time taken for mapping computation in **(ms)**

Algorithm	20	36	64	121	225
coolMap	6.8	7.8	14.8	54.6	146
spreadMap	6.6	7	16.2	49.8	126
hybrid	7.2	10	16.4	67.6	178
greedy	8.2	12.8	26.8	101.4	337.4
checkered	5.8	7.8	17.4	35.8	96
pinned	6.2	7.1	11.4	29.4	92
QP-Best	78.8	347.2	1796.2	19328.8	381342.6

B. Hardware Validation Results

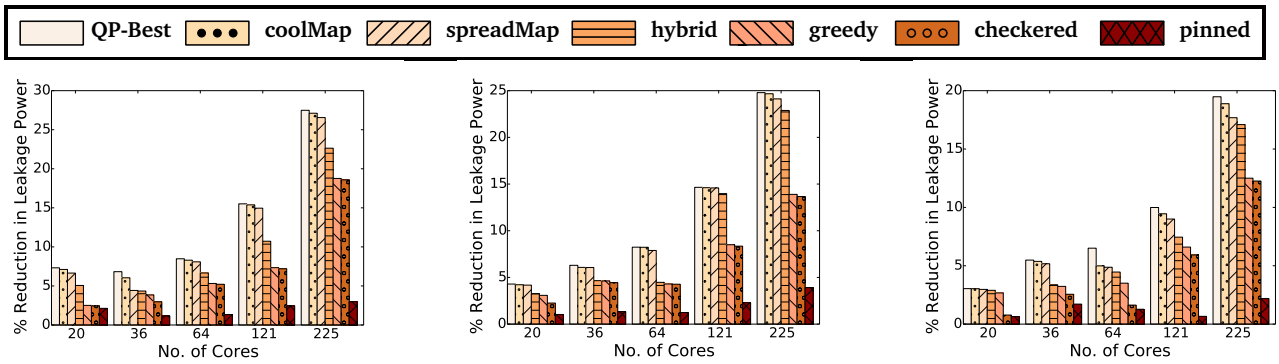
The simulation results for 36 cores showed a maximum savings of 7.11% . But we observe that all our heuristics perform more effectively on a real hardware. The best heuristic (`coolMap`) roughly saves upto $\mathbf{18}\%$(for 25% activity) and $\mathbf{11.5}\%$(for 50% activity) of leakage power consumption. Figures[5a and 5b] show the average % savings in leakage power obtained on `TILE-Gx36™` processor across 20 different multiprogrammed workload combinations that include both heterogenous and homogenous workloads. These results clearly validate the effectiveness and fidelity of the proposed heuristics. Relative performance of various algorithms show an excellent consistency in simulation and hardware.

VII. Conclusion

This paper addresses the problem of minimizing leakage power consumption in large multicore platforms. This work proposes a family of heat spread aware mapping algorithms that compute near-optimal task to core mappings and achieves substantial leakage power savings of 24.67%(average) and 27.12%(maximum) for a variety of workloads. The heuristics are able to closely match the results from QCQP based optimal solution. Further, leakage power consumption directly depends on temperature and heat spread due to other tasks running on neighbouring cores impacts the same. This work then presents an experimental evaluation of the proposed algorithms on a real multicore platform. The results clearly show consistent trends between simulation and hardware in terms of relative effectiveness of algorithms on leakage optimization.

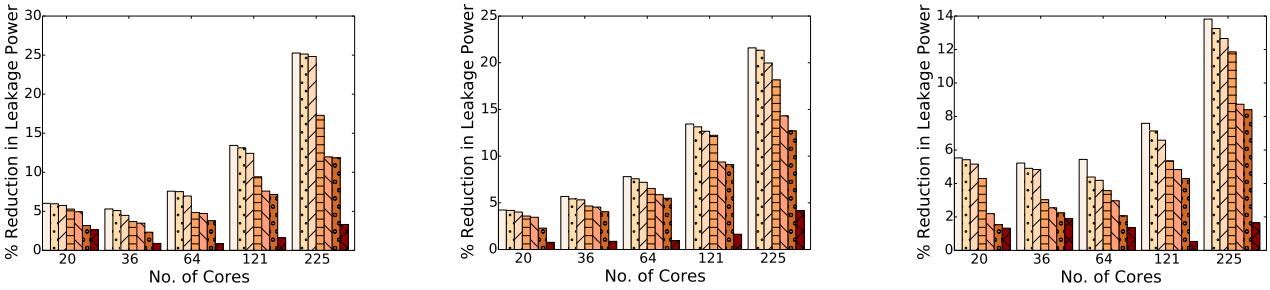
References

[1] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The impact of technology scaling on lifetime reliability," in *DSN*, 2004, p. 177.



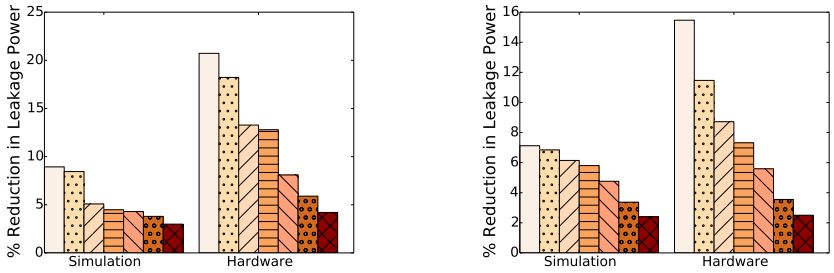
(a) 25% Utilization (b) 50% Utilization (c) 75% Utilization

Figure 3: Workload mix consisting of *lbm, gobmk, sphinx3, omnetpp and mcf*



(a) 25% Utilization (b) 50% Utilization (c) 75% Utilization

Figure 4: Average savings across all simulation workloads *(25 benchmarks from SPEC 2006)*



(a) 25% Utilization (b) 50% Utilization

Figure 5: Average leakage power reduction across all workloads for *36 cores*

[2] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of Multicore Scaling," in *ISCA*, 2011, pp. 365–376.

[3] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," in *DAC*, 2015, pp. 119:1–119:6.

[4] V. Hanumaiah, R. Rao, S. B. K. Vrudhula, and K. S. Chatha, "Throughput Optimal Task Allocation under Thermal Constraints for Multi-core Processors," in *DAC*, 2009.

[5] "Ansys Icepak, "14.0 documentation," ANSYS Inc, 2012."

[6] S. Sarangi, G. Ananthanarayanan, and M. Balakrishnan, "Lightsim: A leakage aware ultrafast temperature simulator," in *ASP-DAC*, 2014, pp. 855–860.

[7] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *DAC*, 2013, pp. 1:1–1:10.

[8] A. K. Coskun, T. T. Rosing, K. Whisnant, and K. C. Gross, "Static and Dynamic Temperature-Aware Scheduling for Multiprocessor SoCs," *IEEE Trans. VLSI Syst.*, vol. 16, 2008.

[9] S. Murali et al., "Temperature Control of High-performance Multi-core Platforms using Convex Optimization," in *DATE*, 2008, pp. 110–115.

[10] T. Ebi, D. Kramer, W. Karl, and J. Henkel, "Economic learning for thermal-aware power budgeting in many-core architectures," in *CODES+ISSS*, 2011, pp. 189–196.

[11] X. Hu et al., "Thermal-sustainable power budgeting for dynamic threading," in *DAC*, 2014, pp. 187:1–187:6.

[12] H. Huang, G. Quan, and J. Fan, "Leakage temperature dependency modeling in system level analysis," in *ISQED*, 2010, pp. 447–452.

[13] "The International Technology Roadmap for Semiconductors (ITRS), 2011."

[14] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Trans. on Arch. and Code Optimization (TACO)*, 2014.

[15] I. Gurobi Optimization, "Gurobi optimizer ref. manual," 2015.

[16] W. Huang, M. R. Stan, K. Sankaranarayanan, R. J. Ribando, and K. Skadron, "Many-core Design from a Thermal Perspective," in *DAC*, 2008, pp. 746–749.