Task Assignment Algorithms for Multicore Platforms with Process Variations

Gayathri Ananthanarayanan[#], Smruti R Sarangi, M Balakrishnan

Abstract– We have moved into an era where modern multicore systems must deal with several critical challenges such as increased power density and high temperatures. In addition to this, ITRS projects that heterogeneity due to manufacturing process variations would continue to rapidly increase in sub-nanometer technology nodes. Spatial heat influence or the lateral heat transfer is becoming more prevalent in manycore systems and has a significant impact on the processor power consumption. Thus it is becoming increasingly difficult to meet the application throughput requirements while adhering to the system power budget and thermal constraints. Many of the existing task mapping schemes that does not take into account the underlying variations in core frequency and leakage power consumption will result in sub-optimal solutions. In this work, we first formulate the mapping problem as an optimization problem and then present a rank-based, process variation and lateral heat transfer aware application mapping algorithm to solve it heuristically. Our extensive experimental results reveal that the proposed algorithm produces near optimal results (average 6% away from optimal) while being orders of magnitude faster (4000x faster for 225 cores).

Keywords– lateral heat conduction, task mapping, process variations, technology scaling, leakage power

1 INTRODUCTION

Aggressive CMOS technology scaling leveraged by chip manufacturers has led to many cores being fabricated on the same chip. The idea behind this was to harness the compute capability of all the cores simultaneously and achieve significant performance improvement. But the continued scaling of CMOS technology has given rise to several critical issues like increased power density, increased leakage power, and consequent increase in operating temperatures. Spatial heat interference or the lateral heat conduction (`heat flow between different cores on a die`) further exacerbates these challenges in modern many-core processors [1,2]. Thus power and thermal constraints prohibit the concurrent use of all the cores at maximum frequency and limits the achievable performance. This phenomenon is known as Dark Silicon [3].

Another critical challenge in sub-nanometer technologies is the increasing fluctuations posed by IC manufacturing process. The feature sizes of current sub-nanometer devices are an ultra small fraction of the wavelength of light used in the lithographic processes. Thus it is very difficult to scale the resolution limit of the manufacturing processes leading to variability in the dimensions of transistors. These variations typically affect the transistor parameters like the threshold voltage (V_{th}) and the channel length (L_{eff}) leading to variability in the transistors power consumption and switching frequency.

There is a complex relationship between threshold voltage, frequency, temperature (T), power, throughput, and the workload assignment. Figure 1 depicts the dependencies between the various parameters. The threshold V_{th} decreases with increase in temperature. Cores with higher V_{th} are less leaky but have high switching times hence will increase the execution time of the task. Cores with lower V_{th} have low switching times but are highly leaky. Because of the limitations in cooling, each core has a different heat removal capacity. Leakage power consumption and core temperature have a cyclic dependency between each other. Thus, the performance and power consumption of the task will now depend upon the core on which it is executed.

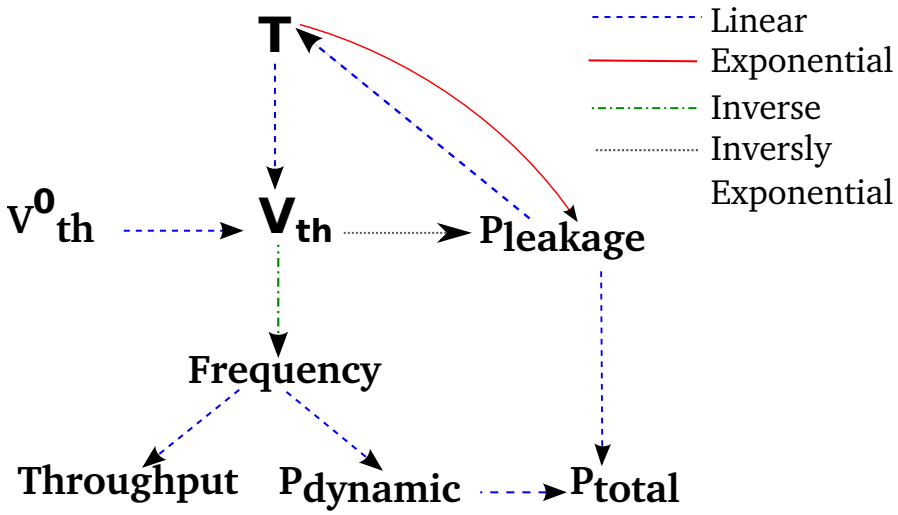


Figure 1: Relationship between Variables

Our aim in this work is to take into account the variations in the core characteristics, the effect of spatial heat interference/thermal coupling and map the tasks of the given workload onto the cores to achieve any of the below mentioned objectives:

- Minimize total power consumption of the chip with a certain guaranteed throughput
- Maximize overall system throughput under a certain power budget
- Compute a set of trade-off solutions in terms of power and performance

2 RELATED WORK

Vast majority of previous works have proposed Dynamic frequency and voltage scaling (DVFS) [4, 5] as the primary technique for power management. Conventional frequency scaling techniques were first adopted at the chip-level [6, 7]. Due to their success in providing substantial power savings, it was gradually adopted at per-core [8], memory subsystem [9] and inter-connect levels [10]. Coordinated schemes combining frequency scaling mechanisms at multiple levels were also proposed [11–13]. Most of these prior works try to only maximize performance per unit of power consumed and do not take into account thermal constraint violations. Our work differs from these as we consider the heterogeneity in core frequency only because of the impact of manufacturing process variations. Existing works of Teodorescu et al. [14], Stamoulis et al. [15] consider the heterogeneity in the die due to process variations and try to maximize the performance of the given set of workloads. There are many reported works that address variation aware task mapping [16–18] that tries to take into account the communication requirements between the tasks while trying to maximize the performance.
Ding et al.'s work [19] talks about adaptive application execution i.e assigning threads of an application to the cores of a chip multiprocessor with process variation using thread mapping schemes that can potentially exploit the existing heterogeneity in power and performance characteristics of the cores to optimize performance, energy consumption and energy-delay product. They also suggest that instead of globally clocking all the cores to the lowest frequency, if a subset of cores are reconfigured to their lowest frequency based on specific application behavior, significant reduction in energy-delay product can be obtained. Meng et al. [20] also proposed a very similar approach of core to thread mapping that reduces the overall power consumption for chip multiprocessors fabricated under extreme parameter variations. But none of the existing works takes into account the effect of spatial thermal coupling or the leakage temperature feedback loop as we do in this work. Also, most of the existing heuristics generate a single approximate solution for a single objective while in our work, we also give a methodology to compute trade-off solutions while considering multiple objectives.

3 BACKGROUND

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

3.1 Models

3.1.1 Architecture Model

We consider a tiled processor architecture consisting of N tiles connected to each other through 2D mesh topology and are organized as a grid of X*Y where $X * Y = N$. Each tile comprises a processor core, L1/L2 cache subsystem together with a memory management unit. In this work, the processor cores are homogeneous with respect to micro-architecture but heterogeneous in terms of performance due to process variations. Each core runs at a different frequency and consumes different amount of leakage power.

3.1.2 Workload Model

We consider a `bag-of-tasks` workload model with 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 SPEC 2006 benchmark suite. More details will follow in Section 6.

3.1.3 Power and Thermal Models

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:

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

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

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

We use the Hankel transform based temperature estimation framework of [2] to model the heat spread function of a point heat source. This model characterizes the lateral heat conduction as the sum of radially symmetric rapidly decaying function and a constant κ. Constant κ captures the effect of the entire die heating up because of a point power source. This happens because some of the heat is transferred to the spreader, which in turn transfers some energy to all the points in the die.

This thermal model 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.

3.1.4 Process Variation Model

The typical transistor parameters that are affected due to process variations are the threshold voltage(V_{th}) and the channel length (L_{eff}). Both these parameters are related to each other by a simple equation proposed by [21]. L_{eff}^0 and V_{th}^0 refer to the nominal values.

$$L_{eff} = L_{eff}^0 \left(1 + \frac{V_{th} - V_{th}^0}{2V_{th}^0}\right) \tag{3}$$

In this work also, we consider only the variation in threshold voltage. We model the systematic and random components of the threshold voltage variation as described in [21]. As leakage power consumption and threshold voltage are related to each other as given by Equation 1, variations in threshold voltage translates to variations in leakage power consumption at ambient temperature. We include these variations in our leakage power model. Similarly, the impact of threshold voltage variations on the peak operating frequency of various cores in the processor is obtained using the alpha power law [21].

$$Frequency \propto \frac{(V - V_{th})^\alpha}{V(1 + V_{th}/V_{th}^0)} \tag{4}$$

Figures 2 and 3 show the threshold voltage and frequency variation maps of a 64-core processor simulated using Varius framework. In Figure 3, there are different possible frequencies in a core block due to variation in individual transistor switching frequencies. But the operating frequency of the core will be the maximum frequency that the slowest transistor can support without suffering from timing faults [21].

3.1.5 Performance Model

We use the concept of CPI stacks [22] to obtain the average number of cycles utilized per instruction (CPI). It breaks the cycles used per instruction into multiple components like baseline cycles, cycles lost due to branch and memory.

$$\begin{aligned} CPI_{tot} &= CPI_{core} + CPI_{L1} + CPI_{L2} + CPI_{dram} \\ CPI_{core} &= BaseCPI + CPI_{branch} \end{aligned} \tag{5}$$

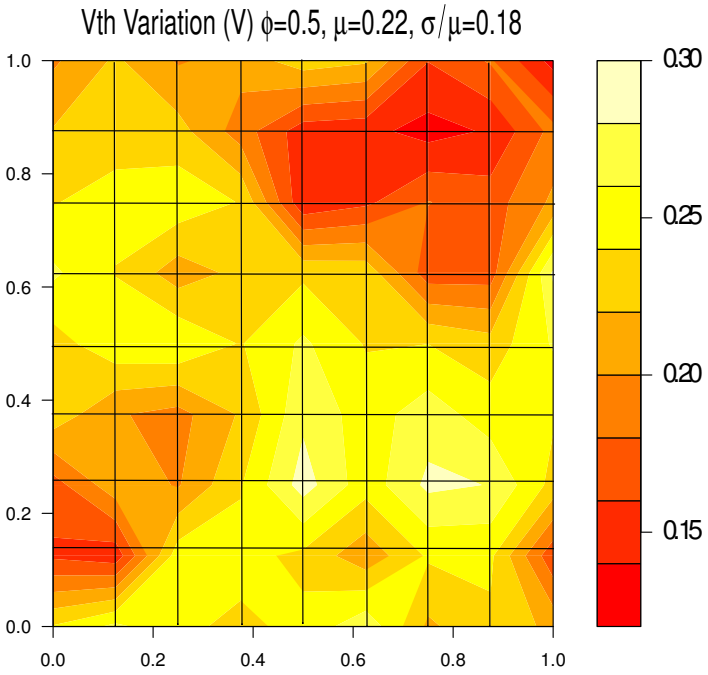


Figure 2: Sample V_{th} variation map of a 64-core processor

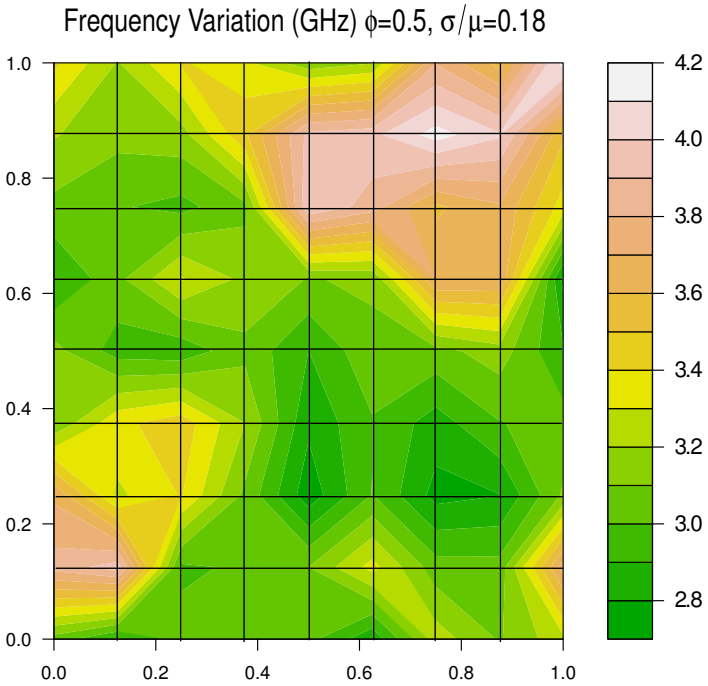


Figure 3: Sample frequency variation map of a 64-core processor

The core, L1 and L2 caches are tied together in a tile in our architecture and runs at the same frequency. To model effects of frequency heterogeneity among the tiles on the performance of the application, we measure the changes in terms of absolute time. We measure the average seconds used (SPI) by the core, L1 and L2 components of the instruction execution (refer Equation 5) at the nominal frequency (F_{nom}) and scale it to obtain SPI for a new frequency $F^{'}$. DRAM access time remains the same irrespective of changes in tile

frequency. The throughput achieved is measured in terms of Instructions per second (IPS).

$$
\begin{aligned}
SPI &= \frac{CPI}{frequency} \\
SPI(F^{'}) &= SPI(F_{nom}) * \frac{F_{nom}}{F'} \\
IPS(F^{'}) &= \frac{1}{SPI(F')}
\end{aligned}
\tag{6}
$$

4 PROBLEM FORMULATION

In every epoch, there are M tasks ($M \leq N$). Let the M tasks be denoted by $\tau_1 \ldots \tau_M$. The average dynamic power consumption of the j^{th} be denoted by τ_j^{DP}.
For each of the N cores indexed by i, Let P_i^d, P_i^s, P_i^{tot} and T_i be the dynamic, leakage, total power and temperature respectively.

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

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

A_{ik} is the thermal resistance co-efficient between core i and k (see Equation 14). Thermal resistance is defined as the difference in the temperature of a core k when a unit of heat energy flows through core i in unit time.The SI units of thermal resistance is Kelvins per Watt. The higher the value of A_{ik}, the smaller the heat transfer and the lower the value of A_{ik}, the greater the heat transfer

4.0.1 Objective Function

The optimization goal in this work is to study the trade-off between the total power consumption and the system throughput achieved. We formulate the mapping problem as a bi-objective weighted sum optimization problem. One component of the objective function ($\sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{P_i^{tot}}$) presents the total sytem power consumption while the other component of the objective function ($\sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{IPS_i}$) presents the system throughput which quantifies the aggregated throughput of all the tasks that are executed in the system. We have two conflicting requirements to satisfy. We aim to increase the system throughput while trying to reduce the total power consumption. For consistency, we transform the maximization objective $\texttt{Max}(\sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{IPS_i})$ into equivalent minimization objective $\texttt{Min}(-\sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{IPS_i})$ to cast the problem as a single objective optimization problem. The problem is now constructed as a sum of objective functions multiplied by weighting coefficients Φ and $1 - \Phi$. Normalization of the two components of the objective function is carried out to ensure the consistency of the optimal solutions.

$$
\texttt{Minimize}(\Phi \cdot \mathcal{K}_1 \cdot \sum_{i=1}^{N} P_i^{tot} - (1 - \Phi) \cdot \mathcal{K}_2 \cdot \sum_{i=1}^{N} IPS_i)
\tag{7}
$$

The trade-offs in the objectives are obtained by varying the coefficient Φ . $\Phi = 0$ leads to throughput maximization mapping while $\Phi = 1$ leads to power consumption minimization mapping. $\mathcal{K}_1$ and $\mathcal{K}_2$ are fixed weights used for normalization.

4.0.2 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.

$$\forall j, \sum_{i=1}^{N} X_{ij} = 1, \qquad \forall i, \sum_{j=1}^{M} X_{ij} \leq 1, \quad X_{ij} \in \{0, 1\} \tag{8}$$

- The throughput of task τ_j running on $core_i$ is given by TPT_{ij} (refer Equations 5 and 6).

$$IPS_i = \sum_{j=1}^{M} X_{ij} TPT_{ij} \tag{9}$$

- We consider the core-to-core variations in frequency and leakage power consumption. We assume all the tiles run at the same supply voltage (V_{dd}).

- The dynamic power of a task running at the nominal frequency F_{nom}, will get scaled appropriately when running on a core with frequency $F^{'}$. Let DP_{ij} denote the dynamic power of j^{th} task when executed on i^{th} core.

$$\begin{aligned} DP_{ij} &= \tau_j^{DP}(F_{nom}) * \frac{F_i}{F_{nom}} \\ P_i^d &= \sum_{j=1}^{M} X_{ij} \tau_j^{DP} \end{aligned} \tag{10}$$

Leakage power's exponential dependence on temperature can be approximated with piece-wise linear models [23]. For the chip operating temperature range of 40°C to 90°C , we assume a linear model of leakage power and is given by

$$P_{leakage} = P_{leakage}^0 + \beta(T - T_{amb}) \tag{11}$$

$P_{leakage}^0$ is the inherent leakage power at ambient temperature and is captured by α. β captures the $\frac{\Delta P_{leakage}}{\Delta T}$ and is a function of the electrical characteristics of the chip like supply voltage, and threshold voltage. Thus, the variation in leakage power of core$_i$ due to threshold voltage variation is modeled by choosing appropriate α and β.

`Power Gating`: Any core that is not dissipating any dynamic power is turned off such that the leakage power dissipation is zero. During any epoch, if there are 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 new equation for leakage power is:

$$P_i^s = \alpha_i + \beta_i T_i \times \left(\sum_j X_{ij} \right) \tag{12}$$

The total power consumption is the sum of dynamic and leakage components and is given by:

$$P_i^{tot} = P_i^d + P_i^s \tag{13}$$

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

$$T = AP \tag{14}$$

Here, A is a symmetric $n \times n$ matrix, T(temperature) and P(power) are $n \times 1$ vectors.

- We set $T_{SL} = 85$°C as the `chip thermal limit` and this is the maximum allowable temperature that can guarantee safe operation of the chip.
Since both the objective function and the constraints contain quadratic terms, we formulate the mapping problem as a `Quadratically Constrained Quadratic Problem (QCQP)` and use a QCQP solver to obtain the optimal solutions (refer Section6).

5 TASK ASSIGNMENT ALGORITHM

The high computational cost of the optimal solution computed by the QCQP necessitates a faster algorithm. We present a process variation and lateral heat influence aware algorithm (Algorithm 1 **VLA_TAA**) in this section. This rank-based task assignment approach provides near optimal results while being order of magnitude faster. The terms used in the algorithm are defined in Table 1.

Table 1: Glossary of terms

Symbol	Definition
$\mathbf{SP}_N$	Array containing the static power consumption of each core at ambient temperature $\mathtt{T_{ambient}}$.
$\mathbf{F}_N$	Array containing the maximum operating frequency of each core.
$\mathbf{T}_N$	Array containing the temperature of each core. T[i] gives the temperature of i^{th} core.
$\mathbf{flp}_N$	`flp[i]` gives the $x-y$ co-ordinates of the i^{th} core on the die
$\mathbf{LHI}_{N\times N}$	Matrix containing the heat transfer contribution of each core on other cores. Each cell of the row gives the amount of heat transferred from `core I` to `core J` when `core I` is applied with 1 watt of power while taking into account the leakage temperature feedback loop. We call this matrix as the `Lateral Heat Influence` matrix.
$\mathbf{DP}_{M\times N}$	Table containing the dynamic power of the tasks scaled according to the frequency of the core. `DP[M,N]` gives the dynamic power consumption of the M^{th} task when executed on N^{th} core.
$\mathbf{TPT}_{M\times N}$	Table containing the throughput (Instructions per second) of the tasks scaled according to the frequency of the core. `TPT[M,N]` gives the throughput in terms of Instructions per second of the M^{th} task when executed on N^{th} core.
$\mathbf{DIST}_{N\times N}$	Matrix containing the pairwise Euclidean distance between the cores. `DIST[I,J]` gives the Euclidean distance between core I and core J.

The proposed algorithm leverages insights from the complex relationship between threshold voltage, temperature, power, and throughput and builds a solution by iteratively mapping tasks onto cores with the aim of achieving the user-defined objective. In the case of trading off multiple objectives, the algorithm computes $\mathcal{K}$ trade-off solutions instead of approximating to a single solution.

The algorithm takes as input the dynamic power (τ^{DP}) and instructions per cycle (τ^{IPC}) of the task at the base frequency f_0, a priority queue *(coreQ)* of unmapped and available cores, a floorplan *(flp)* and a user defined objective. We consider three different objectives in this work

1. minimizing power consumption with a predefined throughput requirement
2. maximizing throughput under a certain power budget
3. computing trade-off solutions in terms of power consumption and throughput.

The algorithm initially sorts the available tasks in *taskQ* according to the selected objective (Lines 2 to 10) and for each task, the procedure **`computeMap`** loops through all the cores in the *coreQ* in the descending order of priority. The priority is computed depending on the desired objective (Lines 1 to 22). For each core in the *coreQ*, the procedure **`computeRanks()`** computes four different ranks namely, `selfLeakage` ($\mathcal{L}^S_{Rank}$), `mutualHeatInfluence` ($\mathcal{H}^M_{Rank}$), `distance` ($\mathcal{D}_{Rank}$) and `frequency` ($\mathcal{F}_{Rank}$) ranks. It uses matrices $\mathtt{SP}_N$, $\mathtt{LHI}_{N\times N}$, $\mathtt{DIST}_{N\times N}$ and $\mathtt{F}_N$ as inputs to compute the ranks.

The `selfLeakage` component captures the variations in the base static power consumption of the cores. Lower the base leakage of a core, higher its rank. The `mutualHeat Influence` component captures the total amount of heat transferred by the core to the other cores in the chip. Lower the total amount of heat transferred by the core, higher is its rank. The distance component is based on the geometric mean of distances of the current core from already mapped cores that are executing tasks on them. The farther the core from the currently active cores, the higher its rank. The $\mathcal{F}_{Rank}$ component captures the inherent variations in the operating frequencies of the core. Higher the frequency of the core, higher its rank. For the objective of maximizing performance, the idea is to aggressively assign the tasks to cores such that the highest throughput

Algorithm 1 VLA Task **A**ssignment **A**lgorithm

Input: $taskQ$, $coreQ$, flp, τ^{DP}, τ^{IPC}, $objective$
Output: $selQ$

taskQ $\leftarrow$ all tasks to be scheduled *coreQ* $\leftarrow$ priority queue of available(`unmapped`) cores *selQ* $\leftarrow$ list of mapped cores $\mathcal{T}_{flag}, \mathcal{P}_{flag}, \mathcal{TP}_{flag} \leftarrow 1$ $\mathcal{T} \leftarrow T_{ambient}$ $G_{Rank} \leftarrow \infty$ *pTaskQ, selQ* $\leftarrow \phi$
if *(*$objective = MinimizePower$*)* **then**
  taskQ $\leftarrow$ `sort` *taskQ* in decreasing order of base DP
end
else if *(*$objective = MaximizePerformance$*)* **then**
  taskQ $\leftarrow$ `sort` *taskQ* in decreasing order of base IPC
end
else if *(*$objective = PowerPerformanceTradeoff$*)* **then**
  taskQ $\leftarrow$ `sort` *taskQ* in decreasing order of base IPC and DP
end
if *(*$objective = (MinimizePower \vee MaximizePerformance)$*)* **then**
  while *taskQ not empty* **do**
    curTask $\leftarrow$ poll head of *taskQ* *selCore* $\leftarrow$ **`computeMap`**($objective$, $coreQ$, flp, τ^{DP}, τ^{IPS})
    selQ $\leftarrow$ *selQ* $\cup$ *selCore*
  end
end
if *(*$objective = PowerPerformanceTradeoff$*)* **then**
  for $k^{'} \leftarrow 1$ **to** $\mathcal{K}$ **do**
    $selQ_{k'} \leftarrow \phi$ **while** *taskQ not empty* **do**
      curTask $\leftarrow$ poll head of *taskQ* $selCore_{k'} \leftarrow$ **`computeMap`**($objective, coreQ, flp, \tau^{DP}, \tau^{IPS}$, *powCoeff*, *perfCoeff*) $selQ_{k'} \leftarrow selQ_{k'} \cup selCore_{k'}$
    end
  end
end

Procedure computeMap( )

if *(objective* $= MinimizePower)$ *;*
then
  for $i \leftarrow 1$ **to** N **do**
    $SP^{amb}[i] \leftarrow$ *getBaseStaticPower(i)* ;
  end
  CoreQ $\leftarrow$ `sort` *coreQ* in increasing order of SP^{amb};
end
else if *(objective* $= MaximizePerformance)$ *;*
then
  for $i \leftarrow 1$ **to** N **do**
    $\mathcal{F}[i] \leftarrow$ *getFrequency(i)* ;
  end
  CoreQ $\leftarrow$ `sort` *coreQ* in decreasing order of $\mathcal{F}$;
end
else if *(objective* $= PowerPerformanceTradeoff)$ *;*
then
  for $i \leftarrow 1$ **to** N **do**
    $SP^{amb}[i] \leftarrow$ *getBaseStaticPower(i)* ;
    $\mathcal{F}[i] \leftarrow$ *getFrequency(i)* ;
  end
  CoreQ $\leftarrow$ `sort` *coreQ* in decreasing order of $\mathcal{F}$ and then in increasing order of SP^{amb};
end
return First element of *CoreQ* if $size(coreQ) = N$;
`computeRanks()` ;
$minG_{Rank} \leftarrow minimum(G_{Rank})$;
$topCore \leftarrow$ core with $minG_{Rank}$ in *coreQ* ;
`updateState()` ;
`checkVitals()` ;
if *(*$\mathcal{T}_{flag} \wedge \mathcal{P}_{flag} \wedge \mathcal{TP}_{flag} \neq 0$*);*
then
  $selCore \leftarrow topCore$;
  `deque` $selCore$ from $coreQ$;
  `updateFlags();`
  return $selCore$;
end
else
  $pTaskQ \leftarrow curTask$;
  `updateFlags();`
  return `Feasible Core not found`
end

Procedure computeRanks( )

1 **if** *(*$objective = (MinimizePower \vee MaximizePerformance)$*) ;*
2 **then**
3 $topQ \leftarrow$ `topmost` **k** cores of *CoreQ* ;
4 **for** $j \leftarrow 1$ **to** k **do**
5 $\mathcal{L}^S[j] \leftarrow$ *getBaseStaticPower(j)* ;
6 $\mathcal{H}^M[j] \leftarrow$ *computeHeatInfluence(j)* ;
7 $\mathcal{D}[j] \leftarrow$ *computeDistance(j,selQ)* ;
8 $\mathcal{L}^S_{Rank}[j] \leftarrow$ *getRank(j,* $\mathcal{L}^S[j]$*)* ;
9 $\mathcal{H}^M_{Rank}[j] \leftarrow$ *getRank(j,* $\mathcal{L}^M[j]$*)* ;
10 $\mathcal{D}_{Rank}[j] \leftarrow$ *getRank(j,* $\mathcal{D}[j]$*)* ;
11 $G_{Rank}[j] \leftarrow \mathcal{L}^S_{Rank}[j]$+$\mathcal{H}^M_{Rank}[j]$+$\mathcal{D}_{Rank}[j]$;
12 **end**
13 **end**
14 **if** *(*$objective = PowerPerformanceTradeoff$*) ;*
15 **then**
16 **for** $j \leftarrow 1$ **to** N **do**
17 $\mathcal{L}^S[j] \leftarrow$ *getBaseStaticPower(j)* ;
18 $\mathcal{H}^M[i] \leftarrow$ *computeHeatInfluence(j)* ;
19 $\mathcal{D}[j] \leftarrow$ *computeDistance(j,selQ)* ;
20 $\mathcal{F}[i] \leftarrow$ *getFrequency(i)* ;
21 $\mathcal{L}^S_{Rank}[j] \leftarrow$ *getRank(j,* $\mathcal{L}^S[j]$*)* ;
22 $\mathcal{H}^M_{Rank}[j] \leftarrow$ *getRank(j,* $\mathcal{L}^M[j]$*)* ;
23 $\mathcal{D}_{Rank}[j] \leftarrow$ *getRank(j,* $\mathcal{D}[j]$*)* ;
24 $\mathcal{F}_{Rank}[j] \leftarrow$ *getRank(j,* $\mathcal{F}[j]$*)* ;
25 $G_{Rank}[j] \leftarrow$ `powCoeff` * ($\mathcal{L}^S_{Rank}[j]$ + $\mathcal{H}^M_{Rank}[j]$ + $\mathcal{D}_{Rank}[j]$) + `perfCoeff` * $\mathcal{F}_{Rank}[j]$
26 **end**
27 **end**

is achieved and then use $\mathcal{L}^S_{Rank}$, $\mathcal{H}^M_{Rank}$ and $\mathcal{D}_{Rank}$ to take care of the possible reduction in power that can be obtained. For computing a set of $\mathcal{K}$ trade-off solutions in terms of power and performance, the procedure computes the ranks for $\mathcal{K}$ different values of `powCoeff` and `perfCoeff`.

Due to high spatial correlation(ϕ) between values of V_{th} in regions of close proximity, less-leaky cores are present in close vicinity to each other. The assumption that mapping high dynamic power tasks to less-leaky cores will result in minimum power consumption does not hold true because of leakage temperature feedback loop. Initially less-leaky cores will gradually become more leaky because more number of less leaky cores in close vicinity are active simultaneously leading to hotspot formation. Since leakage power has an exponential dependence on temperature, the increase in leakage power will be very significant. Similarly mapping high IPC tasks to high frequency cores will help in maximizing the performance but will also lead to very high power consumption because of the same reason as mentioned above.

After the ranks are computed, the core with the minimum global Rank *(G_{Rank})* is chosen for mapping the current task. The function **`updateState()`** computes the augmented temperature field by scaling the lateral heat interference (`LHI`) matrix with the dynamic power of the current task. The resultant core temperature is calculated using the temperature estimation framework described in [2].

Procedure updateState( )

```
1 C ← selCore ;
2 Update LHI[C] by scaling with DP[curTask] ;
3 Update Global LHI matrix of the chip ;
4 for i ← 1 to N do
5 |   T[i] ← computeTemp(i) ;
6 end
7 Update T_N array ;
```

The function **`checkVitals()`** checks whether the maximum power budget, required throughput and maximum temperature conditions are violated and raises the appropriate flag ($\mathcal{T}_{flag}$ / $\mathcal{P}_{flag}$ / $\mathcal{TP}_{flag}$). If the current task is found to violate any of the requirements, then it is moved to the pending task queue (*pTaskQ*). The algorithm continues to iterate over the remaining tasks in the *taskQ*. When an executing task exits, the algorithm checks whether any valid mapping can be found for the tasks in the pending task queue *pTaskQ*.

Procedure checkVitals( )

```
1 for i ← 1 to N do
2 |   if (T[i] > T_SL);
3 |   then
4 |   |   T_flag ← 0;
5 |   end
6 end
7 if (P^Tot > P^budget);
8 then
9 |   P_flag ← 0;
10 end
11 if (IPS[curTask] < IPS_req);
12 then
13 |   TP_flag ← 0;
14 end
```

6 EVALUATION

To showcase the effectiveness of the proposed variation and lateral heat influence aware algorithm, we compare the performance per watt obtained through our algorithm against the following different solutions.

- **`QCQP`** - Optimal solution computed by the QCQP solver
- **`Checkered`** - Maps the task to an idle core such that active cores and inactive cores are interleaved in a checkerboard pattern.
- **`Greedy`** - Maps the task such that it maximizes the distance to all other active cores.
- **`Pinned`** - Maps the task to the first available idle core in an indexed linear circular list.
- **`Random`** - Maps the tasks on to the cores randomly.
- **`Teodorescu et al. [14]`** - A technique that takes into account, the effect of process variation but does not consider the effect of lateral heat influence.
- **`CoolMap`** - A technique proposed in [1] that takes into account, the effect of lateral heat influence but does not consider process variations.

We first describe our experimental setup and simulation environment. We then evaluate the efficiency of the proposed algorithm.

6.1 Experimental Setup

Our simulation framework consists of Sniper 6.0 architectural simulator integrated with McPAT [24] and the temperature estimation framework from [2]. For our simulations, we consider a wide range of floorplans from 20 tiles (5x4) to 225 tiles (15x15). The baseline chip that we consider in this work is a 20-tile processor manufactured using 32nm CMOS technology. We assume the die area to be constant (400 mm^2) as per the 2011 ITRS report [25]. Table 2 summarizes configuration of the baseline processor.

Table 2: Baseline Architectural parameters

Parameter	**Value**	**Parameter**	**Value**
Cores	20	Technology	32 nm
Frequency	2.66 GHz	V_{dd}	1.0V
L1 I-cache, D-cache			
Associativity	8	Size	32 KB
L2 Cache			
Associativity	8	Size	512 KB

We use Sniper 6.0 architectural simulator integrated with McPAT to obtain the power traces and CPI stacks of the SPEC 2006 bechmarks for the baseline processor (refer Table 2). We use the scaling methodologies and factors described in [3, 25, 26] to model the technology scaling effects and obtain the dynamic power, leakage power, frequency and threshold values for other technology nodes accordingly. We quantify and include the leakage temperature trends for all the technology nodes using MASTAR 5.0.51 [25]. We use the framework described in [2] with the configuration mentioned in Table 3 to compute the leakage converged Green's functions (center, edge, corner) for a chip and obtain temperature values of the tiles. We use the same tool to generate the lateral heat influence matrix as well. To calculate final `critical dimension` CD variability from ITRS roadmap, we need to integrate many entries from the CD tolerance data given by the ITRS tables. We use the methodology described in [27] to calculate the magnitudes of variability from ITRS projections for different technology nodes. [27] uses entries such as Line width roughness, overlay, CD

Table 3: 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

uniformity, CD linearity and CD mean to target from the *Lithography* section of the ITRS tables to calculate the overall variability. Figure 4 gives the complete simulation tool flow.

We present the scaled values for all the technology nodes in Table 4. The values in the columns of Frequency, V_{dd} and Power in Table 4 are normalized values (normalized with 32nm as the base) while the other columns represent absolute values.

Table 4: Scaling Projections

Tech (nm)	No.of cores	Frequency	V_{dd}	Power	V_{th} (V)	Var ($\%\frac{\sigma}{\mu}$)
32	20	1.00	1.00	1.00	0.285	15
22	32	1.31	0.95	0.73	0.220	18
16	64	1.69	0.93	0.55	0.175	28
11	121	2.16	0.90	0.41	0.179	34
8	225	2.72	0.90	0.31	0.186	38

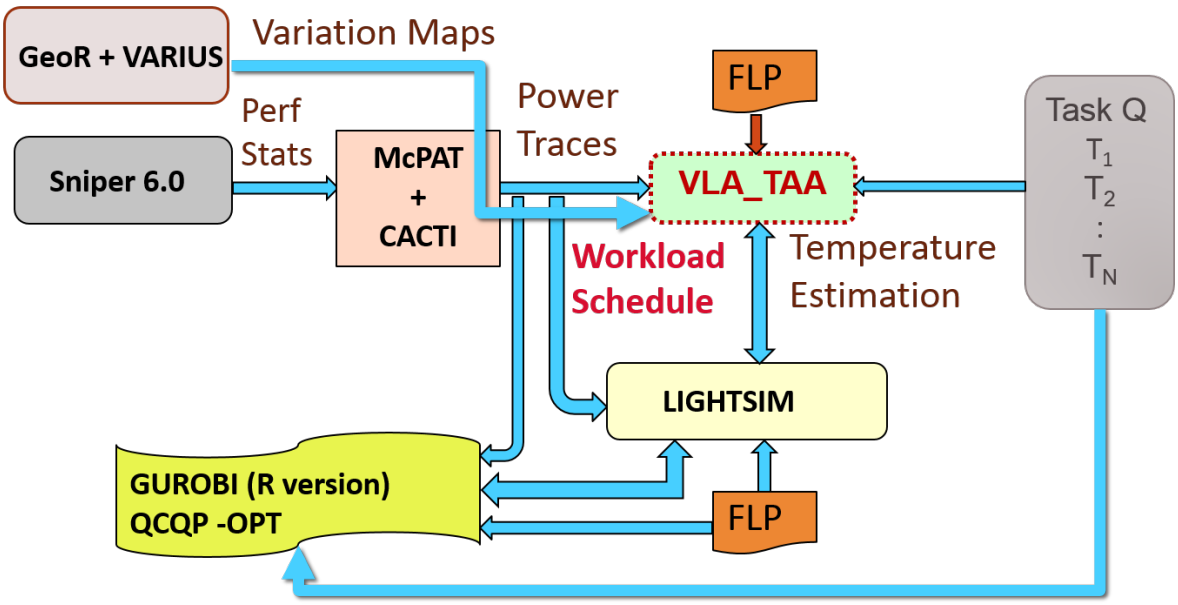


Figure 4: Complete simulation tool chain flow diagram

6.2 Simulation Methodology

In this work, we use applications from the SPEC2006 benchmark suite. We consider a `bag-of-tasks` model, and assume that different cores in the processor run independent tasks. We consider 3 utilization scenarios for evaluation, with 25%, 50% and 75% of the cores active and running tasks. We compose multi-program workload mixes of SPEC2006 applications as indicated in Table 5. For 25% utilization scenario, we generate task queue by combining multiple instances of these mixes. We replicate the instances by **`2x`** and **`3x`** for the 50% and 75% active core scenarios, respectively.

Table 5: Simulation Workload

Workload	Benchmarks
Mix1	hmmer namd tonto gamess h264ref
Mix2	sjeng calculix astar povray gromacs
Mix3	mcf leslie3d libquantum xalancbmk zeusmp
Mix4	GemsFDTD cactusADM milc gcc bwaves
Mix5	lbm gobmk sphinx3 omnetpp mcf

For a given utilization scenario and a workload mix, we identify the baseline power budget and baseline throughput by simulating that workload mix for that utilization scenario for 250 random thread to core mappings. We set the median power of the 250 mappings to be the baseline power budget and median throughput to be the baseline throughput. We vary the power budget and throughput from 80% to 100% of the baseline to explore different constraint scenarios. Previous works [28] and [15] have also used similar approach for generating baseline budgets.

For each utilization scenario, we compare the performance per watt obtained through our algorithm against the different solutions mentioned in Section 6. For computing the optimal solutions of Quadratic Constrained Quadratic Program `(QCQP)`, we use the R interface of Gurobi Mixed Integer Programming solver [29].

Simulation of Process Variations

To simulate process variations, we generated sample variation maps in R using the geoR package as described in [21] and superimposed the floorplan on it using the approach mentioned in [21]. A sample variation map is shown in Figure 3. The σ/μ variation for different technology nodes is given in Table 4. The spatial correlation factor is assumed to be 0.5. This means that a value at a certain point is correlated with values in a radius equal to half the width of the die. We generated 100 sample dies for every technology node and the corresponding results are presented in Section 7.

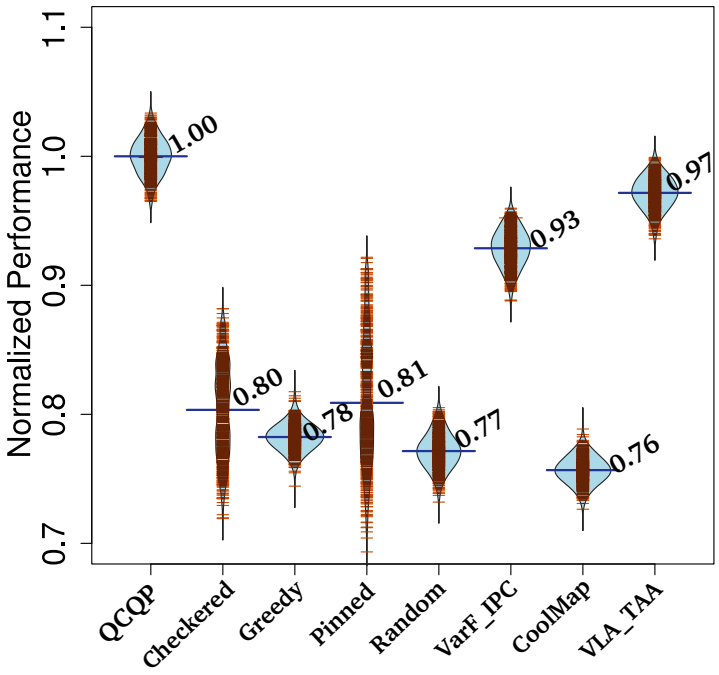
7 RESULTS

On 100 random dies, and for 5 different workload combinations, we evaluate our algorithm against the algorithms mentioned in Section 6. `Performance Per Watt` $\left(\frac{IPS}{Watt}\right)$ represents the energy efficiency metric in this work. We consider the optimal solution computed by the QCQP to be the baseline. All the results presented in the following sections are normalized with respect to the baseline.

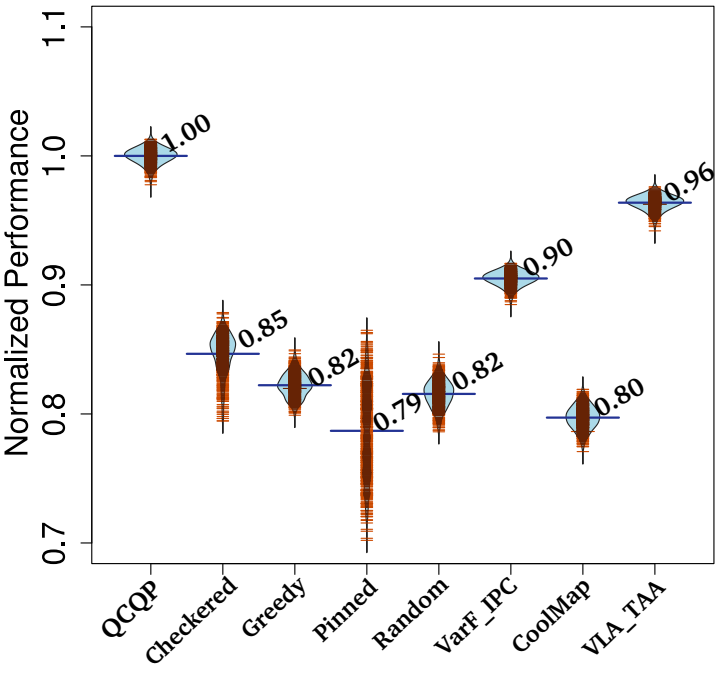
7.1 Maximizing Performance

For the objective of `maximizing performance`, we tabulate (Table 6) the, normalized net system throughput values. For the simulation results presented here, we fixed the chip power budget to be 90% of the baseline power budget (refer Section 6.2). For single threaded applications, instructions per second (`IPS`) is a clear indicator of performance. For multiprogrammed workloads like the ones we use in this work, `IPS` summed over all the applications indicates the net system throughput. Previous works such as [8] and [28] use this as their performance metric.

For each workload, the average net system throughput (across 100 sample dies) obtained by the QCQP solution is considered as the baseline. The net system throughput values obtained by all the other algorithms (for all the 100 sample dies) are normalized with respect to this baseline. We use 5 different multiprogrammed workloads (refer Table 5) in this work.



(a) 25% Utilization



(b) 50% Utilization

Figure 5: Normalized Performance across all Workloads (for 225 cores) for the objective of maximizing throughput under power constraints

7.1.1 Normalized Performance

In Figures[5a and 5b], for 225 cores, for each of the two different utilization scenarios (25% and 50 %), we show the normalized net system throughput across all the 5 different workload mixes for all the 100 sample

dies in two different beanplots[1]. In addition to `net system throughput` values, we also report the `performance per watt values`. For 75% utilization, we observe trends similar to that of 25% and 50% utilization.

Focusing on the performance results in Table 6 and Figures [5a and 5b], we observe that `VLA_TAA` incurs a maximum error of 5% (for 25% activity) and 3% (for 50 and 75%) activity. We also observe the following trends from the simulation results. First, the performance of `VarF_IPC` [14] decreases with increase in utilization. The algorithm tries to map high IPC tasks to high frequency cores that are highly leaky and thus the power budget quota gets exhausted quickly and thus while using this algorithm, the number of unmapped tasks for a given bag-of-tasks increases with increase in activity.

For the case of `Pinned` scheduler, it just maps the task to the first available idle core in an indexed linear circular list without taking into account the underlying process variations and thus for some dies, its performance may be better and in some it may not. But the main limitation of this algorithm is that, since it maps the tasks in the cores one next to the other, there is an increase in power consumption due to hotspot formation. `Greedy` and `CoolMap` are spatial heat interference minimizing algorithms. We can see from the results in Table 6 that the performance of these algorithms improves with increase in activity. One reason is that as the number of free cores decreases with increase in activity, the difference in the normalized net system throughput also decreases.

7.1.2 Normalized Performance Per Watt

We observe from Table 7 and Figures[6a, 6b and 6c] that `VLA_TAA` outperforms QCQP in terms of performance per watt. This is because the QCQP's objective function only aims to maximize throughput while staying within power budget. In contrast, `VLA_TAA` takes into account both the throughput as well as effect of lateral heat interference while computing the global rank (G_{Rank}) metric. This is because for a small degradation in throughput, a significant amount of power can be saved (refer Section 1). As mentioned previously, `VarF_AppIPC` [14] maps the high IPC tasks to high frequency cores, but since it does not take into account the spatial heat interference, it leads to high power consumption. Consequently, its performance per watt metric is much lower than the competing algorithms. `Greedy` and `CoolMap` approaches aims to reduce the total power consumption and thus for higher utilization scenarios, their performance per watt metric is better than `VarF_AppIPC` and `Pinned`.

7.2 Minimizing Power

7.2.1 Normalized Power Consumption

Figures 7a and 7b plot the normalized power consumption of the workload mixes obtained under `VLA_TAA` and various other algorithms for the case of 225 cores and for three different utilization scenarios and Table 8 which tabulates the normalized average power consumption of the workload mixes for all the other core configurations and for all the three different utilization scenarios. For these simulations, we fix the minimum required net system throughput to be 90% of the baseline throughput (refer Section 6.2).

We observe that in the case of 225 cores, for the objective of minimizing power consumption under predefined required throughput, the difference between the optimal solution and `VLA_TAA` is on an average 6% across all the utilization scenarios and workloads. In this case, QCQP outperforms `VLA_TAA` in reducing power consumption as well as in achieving better performance per watt. `VarP_AppP` algorithm maps high dynamic power tasks to less leaky cores. There is a good amount of spatial correlation between values of V_{th} in regions with close proximity. This correlation is modelled as an approximately linear function of the distance between two points, and it is mostly independent of the direction. This thus places less leaky cores in close vicinity. Even though the cores are initially less leaky, but due to lateral heat flow and leakage power-temperature feedback they become more leaky as more and more tasks are placed in close proximity. Thus,

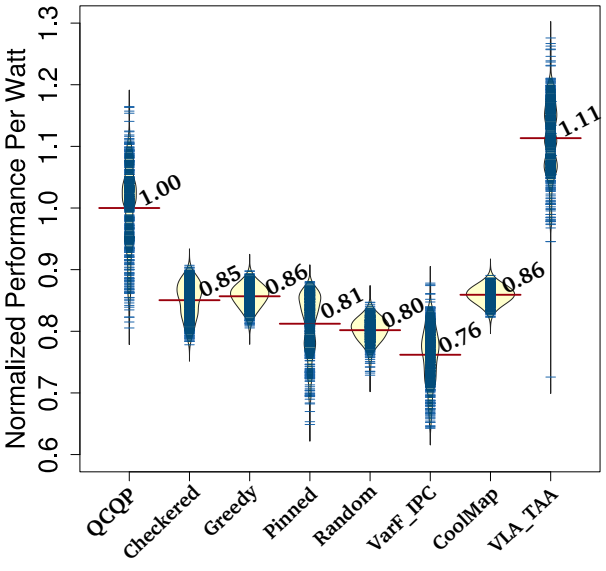
[1]A beanplot is a plot which draws one bean per group of data. A bean consists of a one-dimensional scatter plot that shows all the individual measurements, its distribution as a density trace and an average line for the distribution

Table 6: Normalized Performance averaged over different Workload mixes for the objective of maximizing throughput under power constraints

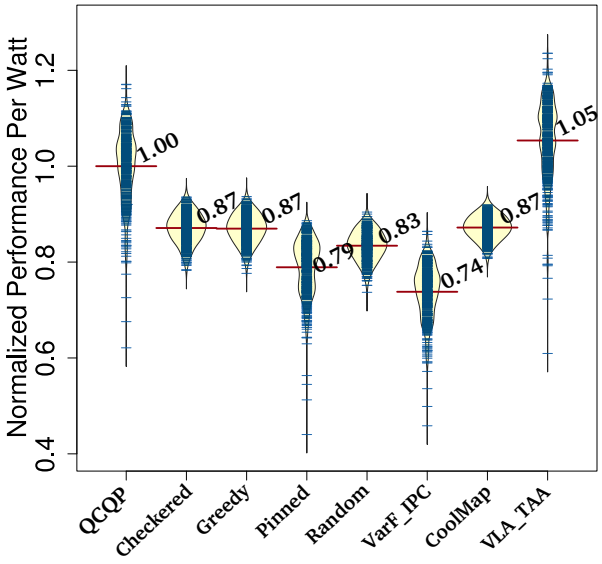
Total No. of Cores	% of Active Cores	QCQP	Checkered	Greedy	Pinned	Random	VarF_IPC Ref [14]	CoolMap	VLA_TAA
20	25	1	0.942	0.922	0.936	0.869	0.975	0.909	0.982
	50	1	0.959	0.936	0.927	0.924	0.972	0.926	0.98
	75	1	0.964	0.949	0.895	0.95	0.968	0.929	0.973
	Average	1	0.955	0.936	0.919	0.915	0.972	0.922	0.978
36	25	1	0.921	0.90	0.921	0.873	0.967	0.837	0.988
	50	1	0.934	0.915	0.882	0.897	0.952	0.86	0.978
	75	1	0.954	0.943	0.851	0.942	0.947	0.893	0.975
	Average	1	0.936	0.919	0.884	0.904	0.955	0.863	0.980
64	25	1	0.894	0.863	0.862	0.892	0.957	0.815	0.982
	50	1	0.924	0.887	0.858	0.908	0.954	0.837	0.977
	75	1	0.947	0.918	0.829	0.945	0.948	0.873	0.966
	Average	1	0.922	0.889	0.849	0.915	0.953	0.842	0.975
121	25	1	0.85	0.832	0.865	0.867	0.942	0.827	0.971
	50	1	0.878	0.897	0.843	0.816	0.934	0.843	0.962
	75	1	0.904	0.923	0.788	0.842	0.911	0.871	0.952
	Average	1	0.876	0.884	0.832	0.842	0.929	0.847	0.962
225	25	1	0.803	0.782	0.810	0.772	0.931	0.748	0.973
	50	1	0.847	0.822	0.792	0.816	0.904	0.794	0.964
	75	1	0.892	0.864	0.754	0.852	0.872	0.802	0.961
	Average	1	0.847	0.823	0.786	0.813	0.902	0.782	0.966

Table 7: Normalized Performance Per Watt averaged over different Workload mixes for the objective of maximizing throughput under power constraints

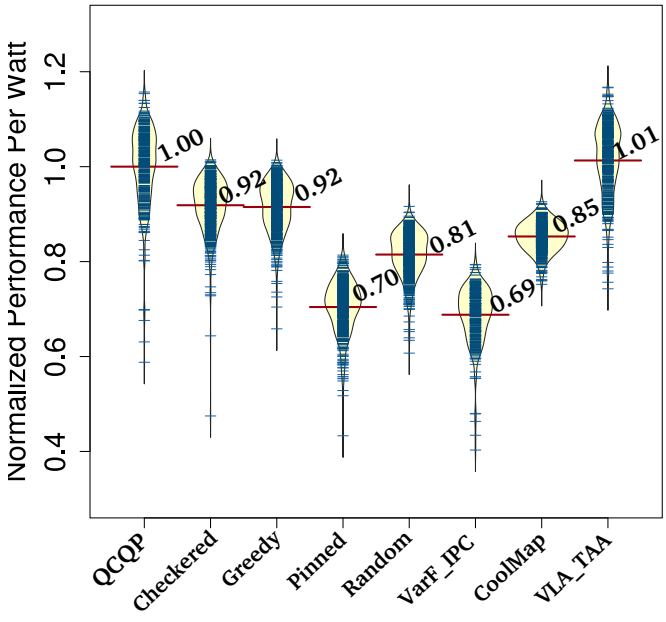
Total No. of Cores	% of Active Cores	QCQP	Checkered	Greedy	Pinned	Random	VarF_IPC Ref [14]	CoolMap	VLA_TAA
20	25	1	1.052	1.124	0.952	0.942	0.926	1.118	1.033
	50	1	1.018	1.034	0.946	0.974	0.905	1.083	1.026
	75	1	1.01	1.01	0.836	0.996	0.90	1.084	1.01
	Average	1	1.03	1.06	0.911	0.971	0.910	1.095	1.02
36	25	1	1.017	1.069	0.923	0.917	0.90	1.032	1.07
	50	1	0.99	1.063	0.916	0.926	0.857	1.037	1.04
	75	1	0.958	1.01	0.902	0.965	0.833	1.023	1.03
	Average	1	0.992	1.05	0.910	0.936	0.864	1.03	1.04
64	25	1	0.945	0.944	0.882	0.943	0.837	0.925	1.05
	50	1	0.962	0.958	0.867	0.947	0.829	0.929	1.03
	75	1	0.962	0.96	0.843	0.970	0.810	0.934	1.02
	Average	1	0.956	0.954	0.860	0.954	0.826	0.93	1.03
121	25	1	0.88	0.907	0.834	0.905	0.795	0.92	1.06
	50	1	0.877	0.975	0.811	0.826	0.773	0.930	1.05
	75	1	0.902	0.941	0.78	0.887	0.72	0.953	1.03
	Average	1	0.887	0.941	0.811	0.873	0.798	0.935	1.05
225	25	1	0.849	0.857	0.811	0.802	0.762	0.86	1.11
	50	1	0.871	0.870	0.790	0.834	0.740	0.87	1.05
	75	1	0.92	0.915	0.70	0.813	0.693	0.854	1.01
	Average	1	0.882	0.884	0.772	0.811	0.760	0.865	1.07



(a) 25% Utilization



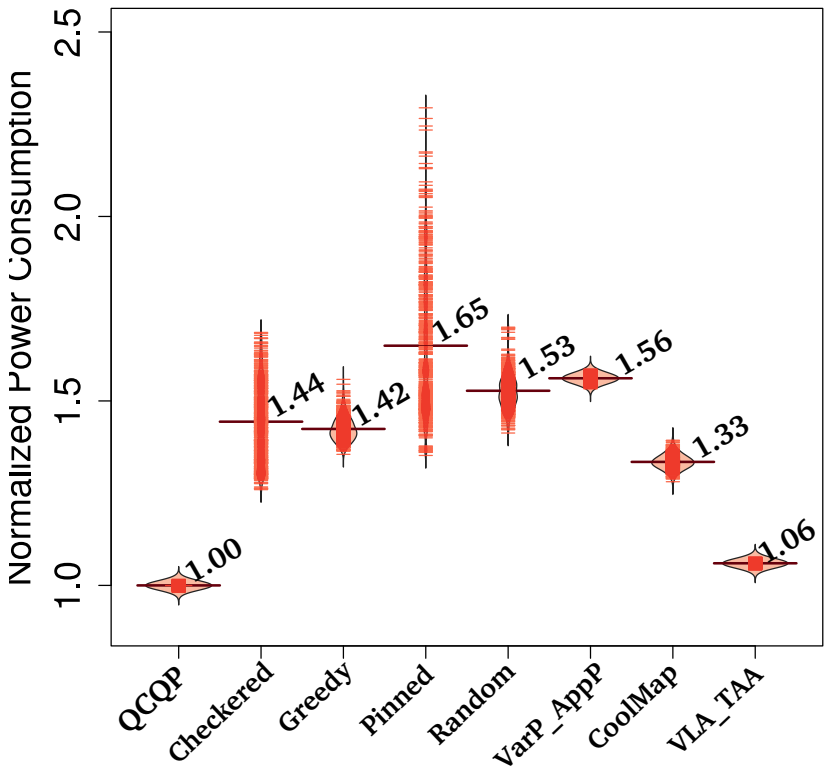
(b) 50% Utilization



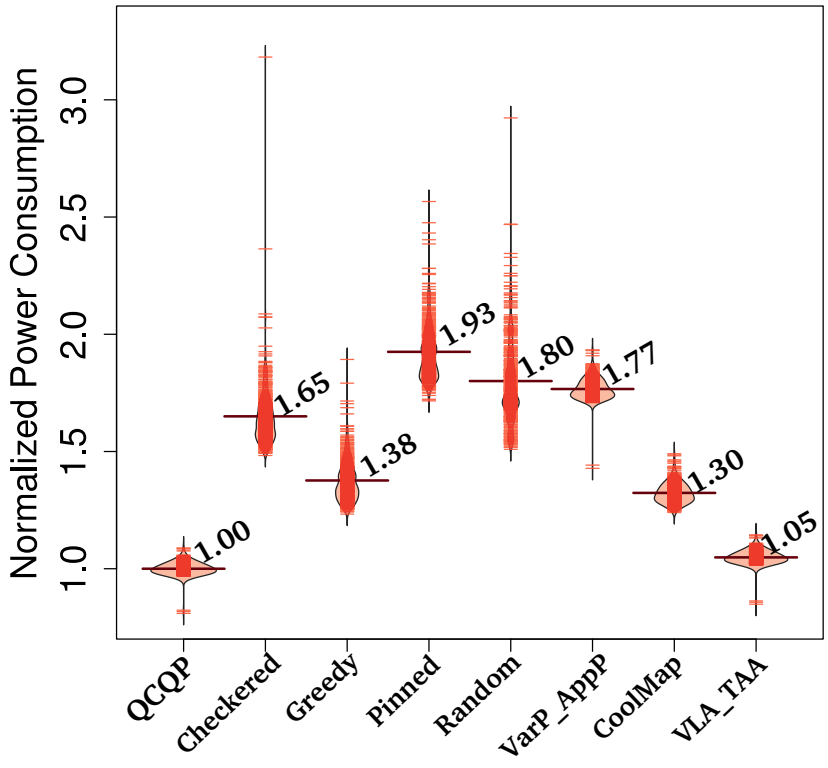
(c) 75% Utilization

Figure 6: Normalized Performance Per Watt averaged across all Workloads (for 225 cores) for the objective of maximizing throughput under power constraints

both `Pinned` and `VarP_AppP` algorithm suffer from hotspot formation and increased power consumption. For some workload mixes, both `Pinned` and `VarP_AppP` algorithm violates the power and thermal constraints inorder to satisfy the predefined required throughput constraint. Even though `checkered` algorithm interleaves active and inactive cores, the heat is spread throughout the die and on an average there is around 52% increase in power consumption (compared to optimal QCQP solution) for 50% and 75% utilization scenarios.



(a) 25% Utilization



(b) 75% Utilization

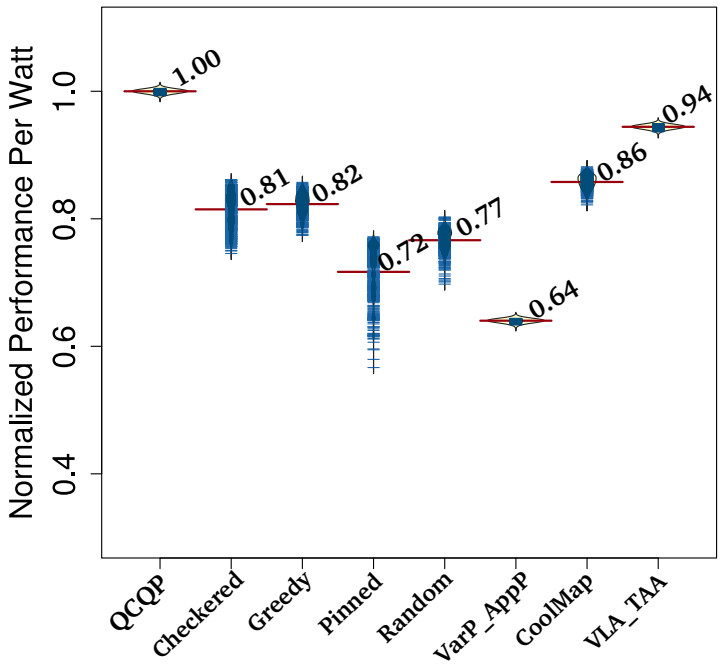
Figure 7: Normalized Power Consumption across all Workloads (for 225 cores) for the objective of minimizing power under pre-defined throughput requirements

Table 8: Normalized Power Consumption averaged over different Workload mixes for the objective of minimizing power under pre-defined throughput requirements

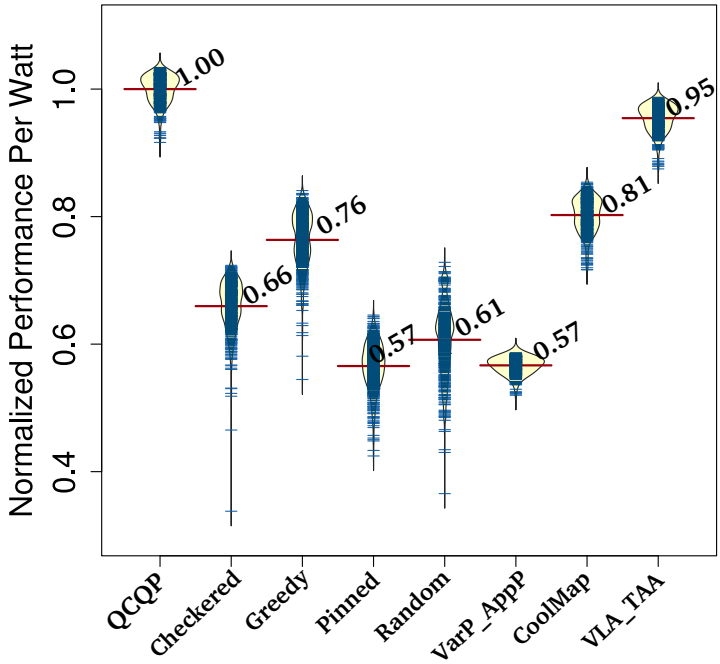
Total No. of Cores	**% of Active Cores**	**QCQP**	**Checkered**	**Greedy**	**Pinned**	**Random**	**VarP_AppP Ref [14]**	**CoolMap**	**VLA_TAA**
20	25	1	1.245	1.217	1.329	1.235	1.283	1.155	1.029
	50	1	1.303	1.207	1.369	1.321	1.320	1.120	1.020
	75	1	1.321	1.172	1.40	1.394	1.348	1.083	1.012
	Average	1	1.290	1.199	1.366	1.316	1.317	1.12	1.02
36	25	1	1.293	1.238	1.371	1.318	1.332	1.165	1.029
	50	1	1.326	1.187	1.425	1.366	1.382	1.138	1.024
	75	1	1.359	1.164	1.461	1.414	1.385	1.109	1.018
	Average	1	1.326	1.196	1.419	1.366	1.366	1.137	1.02
64	25	1	1.357	1.246	1.445	1.397	1.417	1.179	1.038
	50	1	1.443	1.178	1.624	1.539	1.440	1.130	1.032
	75	1	1.462	1.142	1.686	1.594	1.496	1.091	1.024
	Average	1	1.420	1.19	1.585	1.51	1.451	1.13	1.03
121	25	1	1.402	1.304	1.498	1.432	1.471	1.232	1.044
	50	1	1.488	1.261	1.668	1.568	1.514	1.197	1.040
	75	1	1.545	1.250	1.721	1.612	1.572	1.184	1.032
	Average	1	1.478	1.272	1.629	1.537	1.519	1.204	1.039
225	25	1	1.444	1.424	1.650	1.528	1.561	1.335	1.053
	50	1	1.515	1.417	1.851	1.663	1.632	1.314	1.047
	75	1	1.650	1.424	1.926	1.801	1.767	1.324	1.041

7.2.2 *Normalized Performance Per Watt*

We present in Table 9 and Figures 8a and 8b the normalized performance per watt obtained for the objective of minimizing power consumption under a certain predefined throughput.



(a) 25% Utilization



(b) 75% Utilization

Figure 8: Normalized Performance Per Watt across all Workloads (for 225 cores) for the objective of minimizing power under pre-defined throughput requirements

Table 9: Normalized Performance Per Watt averaged over different Workload mixes for the objective of minimizing power under pre-defined throughput requirements

Total No. of Cores	% of Active Cores	QCQP	Checkered	Greedy	Pinned	Random	VarP_AppP Ref [14]	CoolMap	VLA_TAA
20	25	1	0.861	0.879	0.805	0.864	0.783	0.914	0.967
	50	1	0.807	0.870	0.769	0.795	0.755	0.926	0.985
	75	1	0.787	0.887	0.743	0.749	0.744	0.945	0.994
	Average	1	0.818	0.879	0.772	0.802	0.760	0.928	0.982
36	25	1	0.844	0.884	0.798	0.831	0.750	0.925	0.976
	50	1	0.805	0.901	0.750	0.782	0.723	0.928	0.979
	75	1	0.770	0.902	0.716	0.742	0.721	0.936	0.985
	Average	1	0.806	0.896	0.755	0.785	0.731	0.930	0.917
64	25	1	0.824	0.896	0.774	0.798	0.705	0.928	0.967
	50	1	0.752	0.922	0.668	0.703	0.694	0.946	0.972
	75	1	0.725	0.928	0.629	0.664	0.668	0.957	0.978
	Average	1	0.767	0.915	0.690	0.722	0.689	0.944	0.972
121	25	1	0.817	0.877	0.768	0.796	0.679	0.905	0.959
	50	1	0.746	0.874	0.663	0.703	0.661	0.904	0.963
	75	1	0.695	0.857	0.623	0.665	0.637	0.889	0.970
	Average	1	0.752	0.869	0.685	0.721	0.659	0.899	0.964
225	25	1	0.813	0.823	0.712	0.766	0.640	0.858	0.944
	50	1	0.741	0.805	0.608	0.674	0.613	0.837	0.946
	75	1	0.657	0.787	0.564	0.602	0.567	0.816	0.954
	Average	1	0.737	0.805	0.628	0.681	0.606	0.837	0.948

7.3 Power Performance Tradeoffs

Our algorithm is capable of computing schedules representing the trade-offs between power consumption and throughput achieved. We show through extensive experimentation (refer Figures 10a and 10b) that our algorithm computes different trade-off solutions under different experimental configurations, which are very close to the solutions (on an average 7% difference) computed by the optimal solver while being orders of magnitude faster. The main idea behind the speedup of our algorithm is the generation of several trade-off solutions in parallel. The size of the trade-off solutions ($\mathcal{K}$) is provided as input by the user. Depending on the value of $\mathcal{K}$, our algorithm generates appropriate `powCoeff` and `perfCoeff` values and computes the tradeoffs. We also consider the quality of solutions generated. Our method of computing `G_Rank` takes care of this. We only consider solutions that are not dominated by another solution. The net system throughput and power consumption percentages plotted in Figure 10a represents the following. For a given workload mix, 100% throughput represents the maximum possible net system throughput for that mix. The corresponding power consumption required for achieving that throughput is represented by 100% power consumption. All the other throughput and power consumption values are percentages of this maximum value. In figure 9, for 225 cores, workload Mix3, 25% utilization we show for different coefficient values, the solutions generated by QCQP solver and and VLA_TAA. `Relative STPT` denotes the net system throughput normalized with respect to the maximum possible net system throughput for that workload. Similarly, `Relative TP` denotes the total power consumption normalized with respect to the power consumption required for achieving maximum net system throughput for that workload.

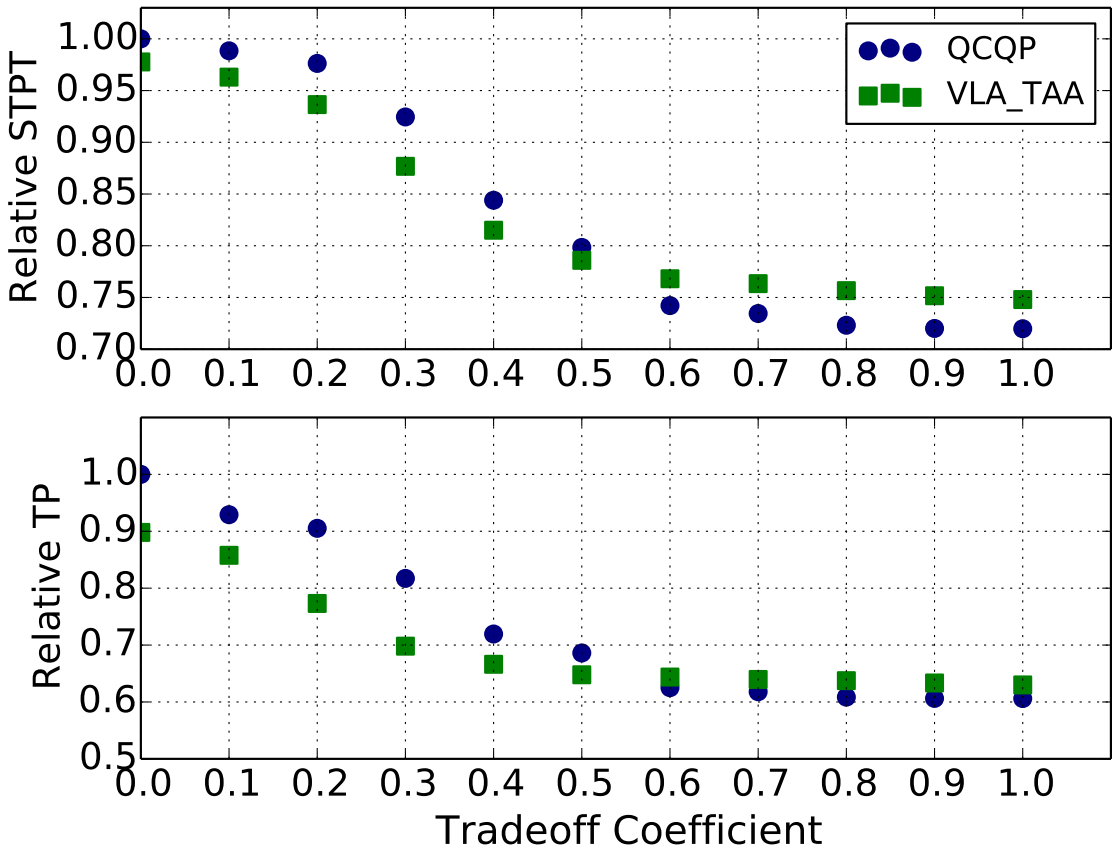
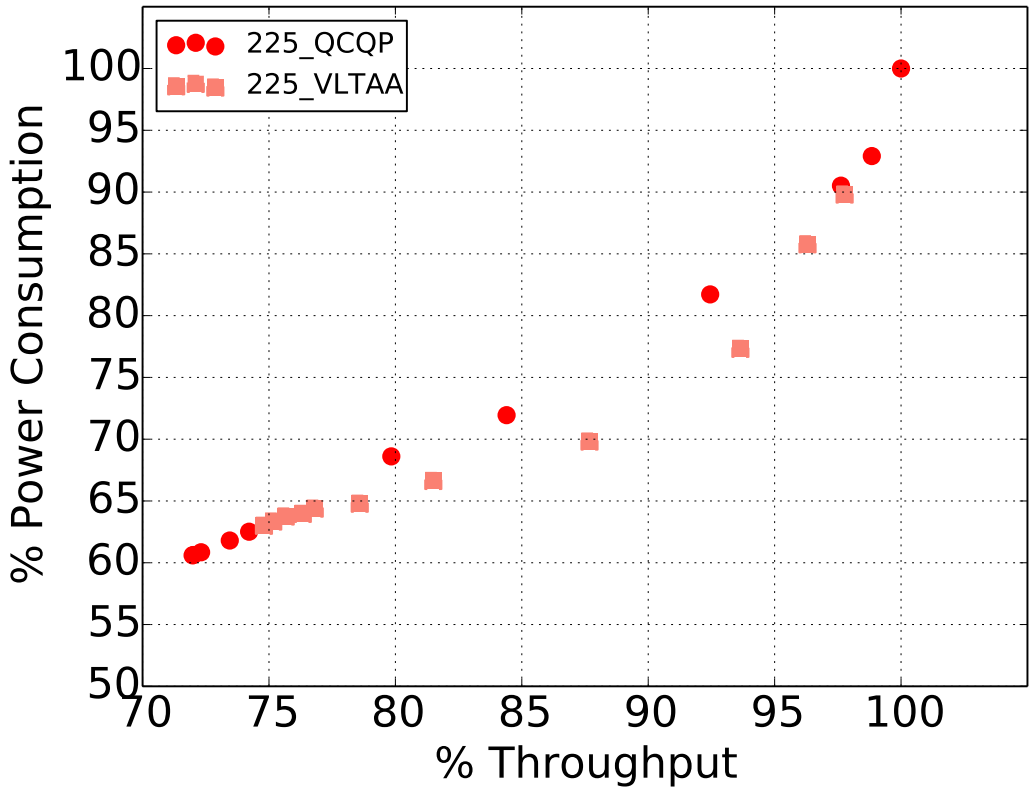
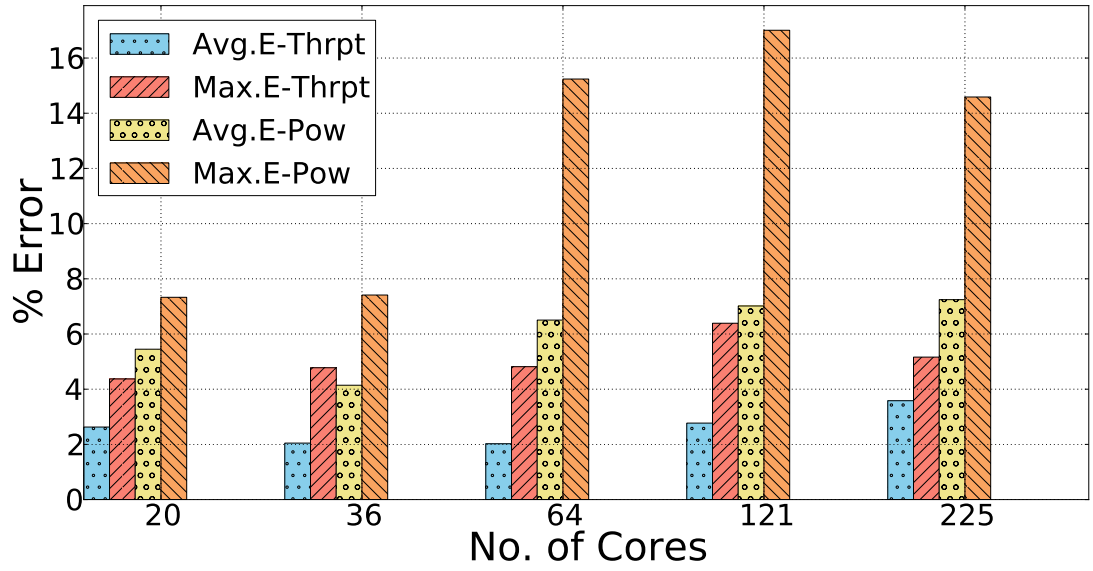


Figure 9: Sample tradeoff solutions



(a) 225 Cores



(b) % Difference between Optimal and VLA_TAA solutions

Figure 10: Power Performance Trade-off comparisons between solutions of Optimal and VLA_TAA Algorithms

7.4 Mapping computation time

For a given bag of tasks, Table 10 shows the time taken by various algorithms to compute the task mapping. We can observe that, for larger number of cores (225), the proposed algorithm *VLTAA* performs 4000x faster than the optimal solver. Even though, the computational overhead of *Pinned* algorithm is very similar to that of *VLA_TAA*, it is clear from the results in Tables 6 to 9, the quality of mappings generated by *VLA_TAA* is better than that of *Pinned* algorithm.

In Table 10, the row **VLA_TAA Speedup** denotes the speedup of VLA_TAA algorithm over the optimal QCQP solver.

Table 10: Time taken for mapping computation in **(ms)**

Algorithm	20	36	64	121	225
VLA_TAA	2.95	5.54	14.04	47.54	92.02
Checkered	3.54	6.69	15.58	50.55	102.58
Greedy	8.30	16.12	39.16	133.29	370.67
Pinned	3.24	5.91	13.82	44.36	93.47
Random	3.32	6.28	16.86	56.16	109.49
Teodorescu et al. [14]	5.04	10.16	25.17	80.7	236.91
CoolMap [1]	4.3	8.26	19.8	64.81	182.24
QCQP	38.39	254.94	1595.95	14526.33	380945.08
VLA_TAA Speedup	13.01	46.01	113.67	305.56	4139.81

8 SUMMARY OF RESULTS AND CONCLUSION

In this work, we proposed an efficient task assignment approach for multicore systems that are affected with process variations. We formulated the mapping problem as a quadratic constraint problem and proposed a heuristic to efficiently solve it. Our work also takes into account the trade-off in multiple conflicting objectives like reducing power consumption as well as maximizing throughput and efficiently generates multiple trade-off solutions at the sametime. The quality of the proposed algorithm was evaluated extensively with 5 different core configurations, 5 different workload mixes and 3 different utilization scenarios. Experimental results indicate that *VLA_TAA* incurs a maximum error of 5% (for 25% activity) and 3% (for 50 and 75%) activity when compared to the optimal results for the case of maximizing the performance under power constraints. Similarly, for the case of minimizing power consumption under predefined throughput requirement, the difference between the optimal solution and *VLA_TAA* is on an average 6% across all the utilization scenarios and workloads.

For the case of maximizing the performance under power constraints,*VLA_TAA* outperforms the QCQP in the performance per watt metric, as VLA_TAA takes into account both the throughput as well as effect of lateral heat interference while computing the global rank. Thus *VLA_TAA* achieves significant power savings for a small degra- dation in throughput.

Our algorithm also computes different trade-off solutions under different experimental configurations, which are very close to the solutions (on an average 7% difference) computed by the optimal solver and order of magnitudes faster. Approximately 4000x speedup is achieved for 225 core configuration.

The technique CoolMap [1] does not take into account the impact of process variations. The performance of CoolMap on an average 11% worser when compared to the performance of all the solutions that take process variations into account. This establishes the need for considering process variations while attempting task mapping algorithms.

Future Work

Another challenge will be to include the effect of dynamic voltage and frequency scaling. Characterizing and modeling the thermal behaviour under various frequencies and utilizing the insights from it while designing DVFS algorithms will improve the effectiveness of the designed algorithm. These approaches will have to take into consideration neighbourhood thermal effects to perform efficiently.

REFERENCES

[1] G. Ananthanarayanan, S. R. Sarangi, and M. Balakrishnan, "Leakage power aware task assignment algorithms for multicore platforms," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 607–612, (**2016**).

[2] S. R. Sarangi, G. Ananthanarayanan, and M. Balakrishnan, "Lightsim: A leakage aware ultrafast temperature simulator," in *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 855–860, (**2014**).

[3] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of Multicore Scaling," in *Proceedings of the 38th annual international symposium on Computer architecture*, ISCA '11, pp. 365–376, (**2011**).

[4] S. S. Jha, W. Heirman, A. Falcón, T. E. Carlson, K. Van Craeynest, J. Tubella, A. González, and L. Eeckhout, "Chrysso: An integrated power manager for constrained many-core processors," in *Proceedings of the 12th ACM International Conference on Computing Frontiers*, CF '15, (**2015**).

[5] K. Ma, X. Li, M. Chen, and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, pp. 449–460, (**2011**).

[6] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proceedings of the 2007 International Symposium on Low Power Electronics and Design*, ISLPED '07, (New York, NY, USA), pp. 38–43, ACM, (**2007**).

[7] S. Kaxiras and M. Martonosi, *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool Publishers, 1st ed., (**2008**).

[8] C. Isci, A. Buyuktosunoglu, C. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-39 2006), 9-13 December 2006, Orlando, Florida, USA*, pp. 347–358, (**2006**).

[9] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ICAC '11, pp. 31–40, (**2011**).

[10] X. Chen, Z. Xu, H. Kim, P. Gratz, J. Hu, M. Kishinevsky, and U. Ogras, "In-network monitoring and control policy for dvfs of cmp networks-on-chip and last level caches," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, pp. 47:1–47:21, (**2013**).

[11] Q. Deng, D. Meisner, A. Bhattacharjee, T. Wenisch, and R. Bianchini, "Coscale: Coordinating cpu and memory system dvfs in server systems," in *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*, pp. 143–154, (**2012**).

[12] X. Chen, Z. Xu, H. Kim, P. V. Gratz, J. Hu, M. Kishinevsky, Ü. Y. Ogras, and R. Z. Ayoub, "Dynamic voltage and frequency scaling for shared resources in multicore processor designs," in *The 50th Annual Design Automation Conference 2013, DAC '13, Austin, TX, USA, May 29 - June 07, 2013*, pp. 114:1–114:7, (**2013**).

[13] D.-C. Juan and D. Marculescu, "Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors," in *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '12, pp. 97–102, (**2012**).

[14] R. Teodorescu and J. Torrellas, "Variation-aware application scheduling and power management for chip multiprocessors," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, pp. 363–374, IEEE Computer Society, (**2008**).

[15] D. Stamoulis and D. Marculescu, "Can we guarantee performance requirements under workload and process variations?," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, ISLPED '16, (textbf2016).

[16] F. Wang, Y. Chen, C. Nicopoulos, X. Wu, Y. Xie, and N. Vijaykrishnan, "Variation-aware task and communication mapping for mpsoc architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, pp. 295–307, Feb (**2011**).

[17] S. Hong, S. H. K. Narayanan, M. Kandemir, and O. Özturk, "Process variation aware thread mapping for chip multiprocessors," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, pp. 821–826, (**2009**).

[18] D. Mirzoyan, B. Akesson, and K. Goossens, "Process-variation-aware mapping of best-effort and real-time streaming applications to mpsocs," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 2s, pp. 61:1–61:24, (**2014**).

[19] Y. Ding, M. Kandemir, M. J. Irwin, and P. Raghavan, "Adapting application mapping to systematic within-die process variations on chip multiprocessors," in *Proceedings of the 4th International Conference on High Performance Embedded Architectures and Compilers*, (**2009**).

[20] K. Meng, F. Huebbers, R. Joseph, and Y. Ismail, "Physical resource matching under power asymmetry." P=ac2 Conf., IBM TJ Watson Research Center, (**2006**).

[21] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "Varius: A model of process variation and resulting timing errors for microarchitects," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 21, no. 1, pp. 3–13, (**2008**).

[22] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith, "A performance counter architecture for computing accurate cpi components," in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, pp. 175–184, ACM, (**2006**).

[23] H. Huang, G. Quan, and J. Fan, "Leakage temperature dependency modeling in system level analysis," in *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, pp. 447–452, March (**2010**).

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

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

[26] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, pp. 119:1–119:6, (**2015**).

[27] M. Orshansky, S. Nassif, and D. Boning, *Design for Manufacturability and Statistical Design: A Constructive Approach*. Springer Publishing Company, Incorporated, 1st ed., (**2010**).

[28] G. Liu, J. Park, and D. Marculescu, "Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pp. 54–61, (**2013**).

[29] I. Gurobi Optimization, "Gurobi optimizer reference manual," (**2015**).