Approximation Knob: Power Capping Meets Energy Efficiency

Anil Kanduri[1], Mohammad-Hashem Haghbayan[1], Amir M. Rahmani[1],
Pasi Liljeberg[1], Axel Jantsch[2], Nikil Dutt[3], and Hannu Tenhunen[1,4]
[1]University of Turku, Finland, [2]TU Wien, Austria
[3]University of California, Irvine, USA, [4]KTH Royal Institute of Technology, Sweden
{spakan, mohhag, amirah, pakrli}@utu.fi, axel.jantsch@tuwien.ac.at, dutt@ics.uci.edu, hannu@kth.se

ABSTRACT

Power Capping techniques are used to restrict power consumption of computer systems to a thermally safe limit. Current many-core systems employ dynamic voltage and frequency scaling (DVFS), power gating (PG) and scheduling methods as actuators for power capping. These knobs are oriented towards power actuation, while the need for performance and energy savings are increasing in the dark silicon era. To address this, we propose approximation (APPX) as another knob for close-looped power management, lending performance and energy efficiency to existing power capping techniques. We use approximation in a pro-active way for long-term performance-energy objectives, complementing the short-term reactive power objectives. We implement an approximation-enabled power management framework, APPEND, that dynamically chooses an application with appropriate level of approximation from a set of variable accuracy implementations. Subject to the system dynamics, our power manager chooses an effective combination of knobs - APPX, DVFS and PG, in a hierarchical way to ensure power capping with performance and energy gains. Our proposed approach yields 1.5× higher throughput, improved latency upto 5×, better performance per energy and dark silicon mitigation compared to state-of-the-art power management techniques over a set of applications ranging from high to no error resilience.

Keywords

Dynamic Power Management; Power Capping; Approximate Computing

1. INTRODUCTION

Multi-core and many-core architectures have become conventional to meet performance requirements of emerging applications. Building denser chips leads to high power density and thermal issues, given the limited cooling solutions. The chip has to function within an upper bound on power consumption, *thermal design power* (TDP) to ensure reliable operation and lower chip temperatures. Power Capping techniques are used to restrict the power consumption below TDP, forcing a section of the chip to be powered off - the powered off area is known as Dark Silicon [5, 20]. Power Capping uses dynamic power management (DPM) techniques such as adaptive Dynamic Voltage and Frequency Scaling (DVFS) [24], Near Threshold Computing (NTC) [25], (Per-core/cluster) Power Gating (PCPG/PG) [13], scheduling techniques like controlled degree of parallelism [12], thread packing [4], task mapping [11] and task migration [15]. We refer to these different means for actuating power consumption as *knobs*. The hierarchy of different power knobs in order of their simplicity and temporal effect on performance is shown in Figure 1(a). These knobs are oriented at power actuation and are effective in power capping, making orthogonal compromises on power and performance. With the amount of simultaneously usable logic decreasing under dark silicon regime, it is necessary to extract higher performance per energy. This creates a need for energy knobs that can offer i) performance and energy gains to complement the existing power knobs, ii) long-term objectives and control.

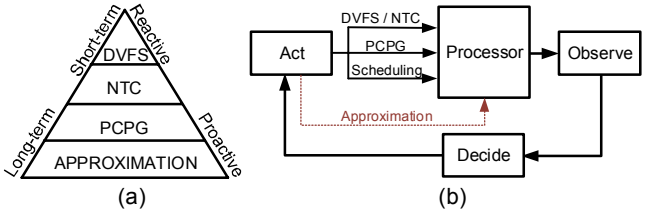

Figure 1: Power Management Knobs

Approximate Computing has emerged as an alternative paradigm that can offer the required performance and energy gains, by trading off some accuracy. Approximation leverages inherent error resilience of applications from several domains such as streaming, image and video processing, machine learning, big data analytics etc. Such applications can tolerate inaccurate results due to their NP-hard and iterative nature, making approximation a better choice to improve performance per energy in the dark silicon era. Recent works on approximation includes techniques such as loop perforation [22], choice of variable accuracy algorithms using logic simplification and task skipping[14, 2, 1], relaxed convergence [9] etc., with approximation at both software and hardware levels. Most of these techniques are open-looped (or closed looped with error tolerance as a constraint), compromising accuracy. In a similar vein, power capping based on DVFS and PG knobs are closed-looped, compromising performance or energy. Putting together the existing power capping techniques with open-looped approximation, we propose a closed-loop power management framework that uses approximation, APPX, as another knob for power capping with performance and energy gains. We use loop perforation and relaxed convergence to provide multiple versions of the same application with different accuracy levels. For evaluation, we used machine learning algorithms which run iteratively on input data sets. Under loop perfo-

ICCAD '16, November 07-10, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2966986.2967002

ration [22], we skip some parts of the input data, reducing the number of iterations, inducing error in the eventual result. The chosen algorithms produce a good enough result with reduced computations, while further iterations try to converge towards an optimal solution. Under relaxed convergence [1], we terminate the computation early, compromising the possibility of a more convergent solution. We present a power management framework that dynamically switches between accuracy levels of the application, choosing from the set of implementations provided. Invocation and actuation of the APPX knob determines the level of accuracy selected. We design a power controller that monitors instantaneous power consumption, workload intensity and utilization metrics of the chip and decides on appropriate actuation of DVFS, PG and APPX in a hierarchical manner. Figure 1(b) shows a top level view of the approximation knob (APPX) in conjunction with other power knobs. While DVFS and PG are used as short-term power knobs, APPX is used as a long-term energy knob, improving performance and energy efficiency, simultaneously ensuring power capping. In this paper, we propose an approximation-enabled power management approach, APPEND, for power capping and throughput enhancement. To the best of our knowledge, ours is the first approach to use approximation in the context of a closed-loop power capping for network-on-chip (NoC) based many-core systems. The contributions of this paper are as follows.

- The approximation knob (APPX) that lends performance and energy gains to power knobs by altering accuracy level of applications from a set of variable accuracy implementations
- A power management framework for power capping in many-core systems with dynamic in-flow of applications using DVFS, PG and APPX knobs
- An accuracy-aware run-time mapping technique that switches between levels of accuracy by mapping approximate tasks or replacing accurate tasks with them
- A run-time control algorithm for power, performance, throughput and energy saving decisions by actuating DVFS, PG and APPX in a hierarchical way.

2. BACKGROUND AND MOTIVATION

Power knobs based on voltage and frequency scaling such as DVFS and NTC are simple, yet effective for power capping due to the dependence of power on V and F. Transistor scaling results in increased leakage power, as the operational voltage approaches threshold voltage (V_t), limiting the scope of DVFS. A special case of voltage scaling is NTC, where supply voltage is intentionally reduced beyond threshold voltage for ultra low power operation. This compromises performance heavily, replacing dark silicon with dim silicon [25]. Power Gating (PG) reduces the static power at the expense of performance, since only fewer cores are simultaneously powered up. DVFS and PG are triggered based on a reactive strategy with short-term objective of power capping, consequently restricting performance and energy gains to a short-term. As opposed to conventional power knobs, approximation provides performance and energy gains for loss of accuracy. We demonstrate the impact of different power knobs on performance of many-core systems using sparse vector multiplication as an example. With applications entering and leaving the system dynamically, performance is determined by *service time* of an application, which is the sum of *wait time* - the time elapsed between application request and starting of the execution and *run-time* - the time consumed in executing the application on chip [12]. Dynamic workload characteristics contribute to

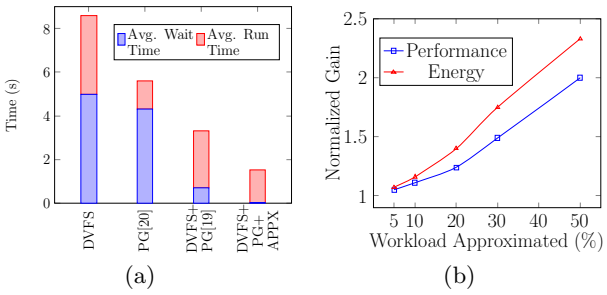

Figure 2: Performance and Accuracy trade-offs with APPX knob. (a) Application Service Time. (b) Accuracy-Energy trade-offs

power violations, forcing actuation of power knobs. We simulate the application for 4 different power knobs viz., DVFS, PG, DVFS+PG [19], and DVFS+PCPG+APPX, using the experimental platform, detailed in Section 5. For APPX, we used loop perforation to skip 10%-50% of input data to generate variable accuracy versions of sparse vector multiplication. The average service time for different knob combinations is shown in Figure 2(a).

In case of using DVFS and PG knobs [7], per-application run-time increases forcing the incoming applications to wait longer, resulting in high service time. The combination of DVFS and PG has relatively better service time using the power management algorithm, as in [19]. With the APPX knob in combination with DVFS and PG, the service time is the lowest, indicating high performance and energy gain within the given power budget. The APPX knob loads applications with relaxed accuracy that have lower workloads and thus low run-time. Consequently, more resources are available for incoming applications, improving the wait-time and the overall service time. Despite effective power capping and possibility of increasing the number of simultaneously active cores, performance still suffers with DVFS and PG when compared to that of APPX. Hence, we propose a hierarchical management for effective combination of these knobs to complement each other. The performance gains of APPX knob come at the expense of accuracy, traded in loop perforation. Figure 2(b) shows the gain in performance and energy for amount of the workload relaxed. It is to be noted that amount of workload traded is not the same as loss in accuracy, which is subjective to input data sets.

3. KNOB ACTUATION SCENARIOS

In our work, we primarily monitor power consumption and workload intensity, along with utilization and network intensities, to make knob actuation decisions. We use accumulated wait-time (AWT) of application requests made, for monitoring the workload. For a set of applications App_1, App_2,..App_N with wait-times w_1, w_2, ...w_N, AWT is:

$$AWT = \sum_{i=1}^{N} w_i \quad (1)$$

The longer an application waits before being serviced, the higher the workload intensity. The power-objective is to restrict the power consumption to below TDP and workload-objective is to restrict the AWT to a parameterizable threshold AWT_{th}. We demonstrate possible scenarios that require knob(s) actuation under diverse power consumption and workload intensities. Figure 3 summarizes these scenarios, representing power consumption, workload intensity and knob actuations employed over a span of execution. Each scenario (a-f) shows power consumption with respect to TDP, applications waiting in the queue, applications that are mapped on the chip with their respective voltage and frequency levels. Voltage and frequency levels of each mapped

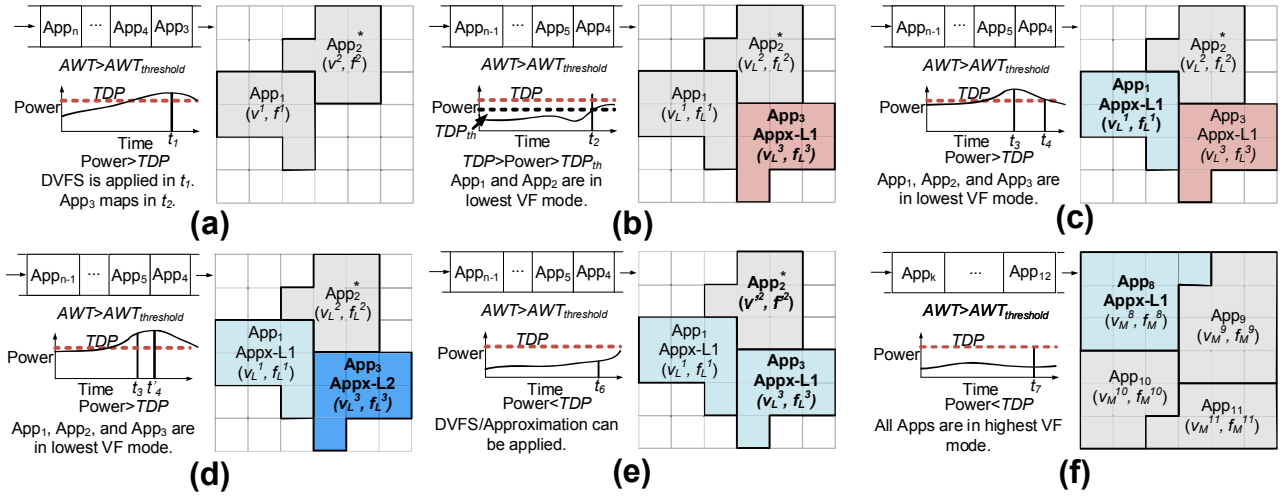

Figure 3: Knob Actuation Scenarios

application (App1, App2,..Appn) are represented as $((v^1,f^1), (v^2,f^2), ...(v^n,f^n))$, with (v_L,f_L) being lowest and (v_M,f_M) being the maximum levels of voltage and frequency respectively. Primary criteria for knob actuation are TDP violation (i.e., power > TDP) and high request rate of incoming applications (i.e., $AWT > AWT_{th}$).

Scenario (a): Two applications App1 (v^1,f^1) and App2 (v^2,f^2) are currently on the system at their respective voltage and frequencies. At time instance t_1, a power violation ($power > TDP$) occurs, along with high request rate ($AWT > AWT_{th}$). The power manager employs the DVFS knob at first to stay within the power budget. The remaining un-occupied cores are power gated using the PG knob. Power-gated cores represent the amount of dark silicon accumulated on the chip. **Scenario (b):** App1 and App2 are now running at their lowest voltage and frequency levels (v_L^1,f_L^1) and (v_L^2,f_L^2) due to triggering of DVFS at t_1. At this stage, power consumption is approaching TDP, indicating a potential violation of power budget. Also, the request rate of applications is high. Anticipating the possibility of TDP violation, the power manager decides to switch the mode of incoming applications to approximate. Since App1 and App2 are already down-scaled in terms of DVFS and remaining cores are power gated, the power manager triggers APPX knob. At time instance t_2, a new application App3 arrives and is hence mapped onto the system in approximate mode as App3-Appx-L1 (shown in plum). As App3 is directly mapped in its approximate version, there is no overhead of mode switching. **Scenario (c):** At time instance t_3, power violation occurs along with a high request rate of applications. All the applications are operating at their lowest possible voltage and frequencies and rest of the cores are power gated. App3 is already being executed at Appx-L1 while App1 is in accurate mode. Since DVFS and PG are used, the power manager now switches the mode of execution of App1, whereas App2 is not approximable. App1 is switched to approximate mode, resulting in App1-Appx-L1, with some overhead (indicated in blue). These actuations may restrict the power within TDP, as shown at t_4. **Scenario (d):** Contradictorily to the previous scenario, the actuations may still not be able to prevent the power violation nor address the high request rate, as shown at instance t'_4. In this case, we further switch the mode of execution to another level of approximation. Since App1 is already executing at Appx-L1, App3-Appx-L1 is switched to App3-Appx-L2, the next level of approximation. This incurs a marginal overhead (indicated in blue) in switching between modes of execution . **Scenario (e):** With preceding knob actuations, power consumption at instance t_6 is well below TDP. This gives enough budget to utilize, prompting an increase in voltage and frequency levels and/or increase in accuracy of computations. We make this a user-defined quality of service (QoS) design parameter, where one can prioritize either throughput or accuracy. Accordingly, either voltage and frequency levels are up-scaled or accuracy level of the approximate application is switched one level ahead. In this scenario, we show App2 being up-scaled to (v'^2,f'^2) and App-3-Appx-L2 being switched to level-1 of approximation, App-3-Appx-L1. **Scenario (f):** Power consumption is well below TDP, leaving ample budget to utilize. This allows cranking up voltage and frequency levels of applications, hence all the applications are operating at their maximum required voltage and frequency levels (v_M,f_M). However, at instance t_7, the request rate is high, although TDP is honored. In this case, to decrease the AWT, App-8 is switched to approximate mode at level-1, App8-Appx-L1.

4. SYSTEM DESIGN

Fulfilling the objectives of power capping while maintaining better throughput requires a power management framework that *monitors* critical chip parameters of power consumption, performance, utilization and network intensity, *decides* on required optimization and *acts* upon the optimization in an observe-decide-act (ODA) loop. We design our power management framework in such an ODA loop fashion for a NoC-based many-core systems. Subject to application requests, the workload, power consumption, utilization, network intensity of the chip varies. Our power management framework monitors these metrics and in case of power violations and higher workload intensities, it decides on actuation of different knobs. The system architecture is shown in Figure 4, and detailed below.

4.1 System Architecture

We present our power management framework for NoC based many-core system that supports dynamic arrival and servicing of applications. We use applications that are modeled as directed task graphs, where each task runs concurrently with other tasks. Each task has its computational intensity and its communication volume with other tasks. Applications are classified as approximable and non-approximable. Approximable applications are those that are modeled as compound task graphs, such that one or more tasks of the application can be replaced by their approximate versions. Incoming applications arrive at the application

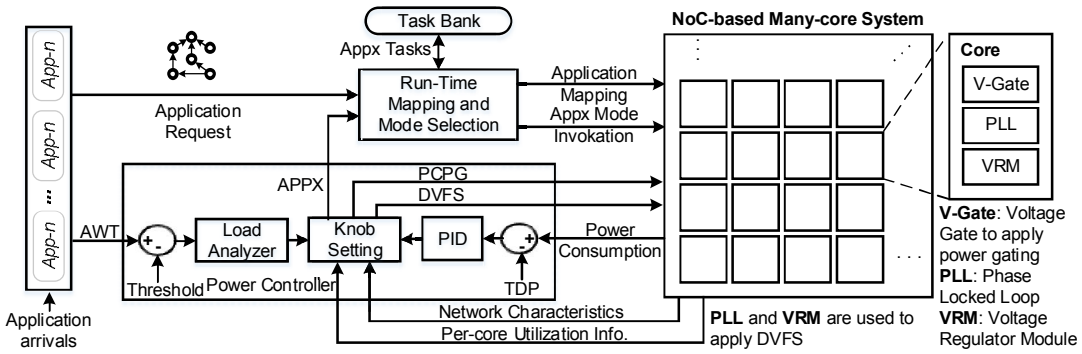

Figure 4: Power Management Framework

repository and make an application request to be serviced. The run-time mapping and mode selection unit (RMSU) is responsible for servicing these requests by allocating on-chip resources per application. RMSU handles application mapping, a one-to-one function of allocating a core-per-task for all tasks of the application. In our framework, we use proactive application mapping *MapPro*, presented in [8]. Parallel execution of multiple incoming applications is possible with RMSU support. RMSU communicates with power controller to send mapping information of current set of applications running on-chip and for actuation decisions, described in sections below.

4.1.1 Power Manager

Power Manager is the central controller that monitors system metrics and decides of actuation of different knobs. Actuation decisions of the power manager are based on per-core power consumption from power meters, per-core utilization from performance counters, network intensity from router buffers, amount of workload and their characteristics from application repository (i.e., approximable or non-approximable). Each core on the network is equipped with per-core power and utilization meters and we trace network intensity as a moving average of packet flow through the buffers at each router. These metrics are sent to the power manager, forming the power monitoring phase. We accumulate wait-time of all the applications that have made an execution request and are currently waiting in the queue to be serviced. We send the accumulated wait-time (AWT), as in Equation 1, to the power manager, forming the workload monitoring phase. We set another parameterizable threshold TDP_{th}, a metric that indicates potential TDP violation, such that $0.66 \times TDP < TDP_{th} < TDP$. Actuation decisions of our power manager for DVFS and PG knobs are based on power controller presented in [19]. DVFS and PG actuations are applied to the chip, as shown in Figure 4. In case of high workload ($AWT > AWT_{th}$), the power manager invokes APPX knob, and chooses the application(s) that can be switched to approximate mode. The difference between AWT and the threshold AWT_{th} determines the level of approximation. The power manager sends the APPX knob invocation, the application chosen to be approximated and the level of approximation to the RMSU. For evaluation, we currently use two levels of approximation in increasing order of accuracy trade-offs. Alternatively, several fine-grained levels of accuracy trade-offs could be used.

Mode Switching: When the current mode of execution is accurate, the RMSU originally maps the accurate version of the task graph. If the application is approximable, the RMSU buffers approximate tasks of the application into the *Task Buffer*. Formulation of the compound task graphs is shown in Figure 5. We generate compound task graphs that include multiple versions of tasks that are approximable, shown in dotted lines. Depending on APPX knob setting,

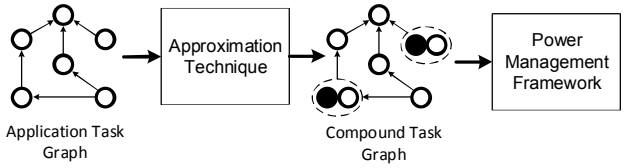

Figure 5: Compound Task Graph - Workflow

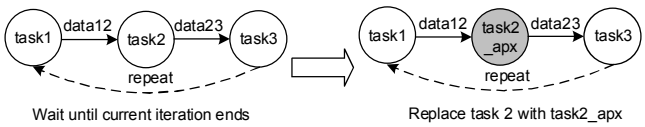

Figure 6: Mode Switching

RMSU chooses the version of task to be included in the application mapping, while the other versions are buffered. With the invocation of APPX knob, there are two possible scenarios for mode switching viz., i) mapping approximate task graphs and ii) switching mode of execution of applications currently running by task replacement. In the former case, the RMSU maps every incoming application in its approximate version by including the approximable tasks instead of accurate tasks, until the mode is switched back to accurate. In the latter case, power manager chooses the application(s) and level of approximation to switch to. Based on these, RMSU identifies the corresponding approximate task from the *Task Buffer* and replaces the accurate task with the approximate task. We modeled applications for evaluation as data dependent concurrent tasks that execute periodically. The computational process repeats until the end of execution with a specified periodicity. Streaming and signal processing applications are good examples which execute periodically over incoming samples of data, where it is possible to relax certain aspects of computation when new data arrives every period. When RMSU has to replace an accurate task with approximate, it lets the current iteration of accurate task's computation to finish execution. It waits until data from the accurate task is received at its destination end task (if any). Once the data transfer is completed, the RMSU loads the approximate task on to the chip, replacing the accurate task. Figure 6 shows the process of task replacement during mode switching. The example has three tasks 1, 2 and 3 out of which task2 is approximable. On invocation of APPX knob, the switching happens in the following sequence. i) The RMSU finds the approximate task *task2_appx* from the *Task Buffer*. ii) It waits until data from task2 (data23) is received at task3. iii) *task2_appx* is loaded by fetching the instruction stream into the cache. iv) After the data is received at task3, the execution of task2 will now start from new instruction stream of *task2_appx*. Depending on size of the instruction cache used, instructions of task2 may require flushing, however this is subject to

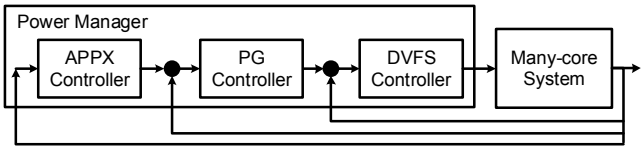

Figure 7: Hierarchy of Knobs

Algorithm 1 Power Management Algorithm

Inputs: P: Instantaneous Power, AWT: Accumulated Waiting Time;
Outputs: $RMSU.inMode$, $RMSU.runMode$, $RMSU.level$: Mode switching commands for RMSU;
Constants: TDP: Power budget, AWT_{th}: AWT Threshold
Global Variables: App_R: Applications currently running, App_X: Applications approximable, App_Y: Applications running in approximate mode;

Body:

1: **while** $AWT \geq AWT_{th}$ **do**
2: **while** $P \geq TDP$ **do**
3: $DVFS_{down}(Apps_R)$;
4: **if** success **then** break;
5: PCPG($cores_{un}$);
6: **if** success **then** break;
RMSU.inMode = APPX;
7: **for** App_X **do**
8: RMSU.runMode(App_X, level-1);
9: **if** (success) **then** break;
10: **for** App_Y **do**
11: RMSU.runMode(App_Y, level-2);
12: **if** success **then** break;
13: **while** $AWT < AWT_{th}$ **do**
14: **while** $P < TDP$ **do**
15: $DVFS_{up}(Apps_R)$;
16: **if** fail **then** break;
17: powerup($cores_{un}$);
18: **if** fail **then** break;
19: RMSU.inMode = ACC;
20: **if** fail **then** break;

hardware platform. Since the computational process of the application is periodic in nature, data is not changed with mode switching and moving the data or flushing the data cache is not needed. The state of the application is hence preserved at the end of the period. It is to be noted that task migration [15] has an appreciable overhead in moving both instructions and data, which is a widely used approach in dynamic power and thermal management. In comparison, the mode switching overhead is lesser, as it involves insertion of new instructions alone and does not need any data accesses. The overhead incurred in mode switching is elaborated in Section 5.

Mode Switching Vs Dynamic Knobs: Hoffman *et al.* [9] have used dynamic knobs that reside in program space to scale accuracy of applications. However, this restricts accuracy trade-offs only to applications with possibility of specific relaxed execution that do not require compile time support. Approximation techniques that use logic simplification and minimization with a different implementation need compile time support, and cannot take advantage of dynamic knobs approach. Mode switching overcomes this limitation, widening the scope to several application domains and approximation techniques. Dynamic knobs can be treated as a best case sub-set of mode switching.

4.2 Power Management Algorithm

We employ the DVFS and PG knobs synergistically with APPX in a hierarchical way, as shown in Figure 7. Triggering and disciplined tuning of these knobs together for power capping is handled by a power management and mode switching algorithms. Our hierarchical power management algorithm is presented in Algorithm 1. In cases of power violation, it first applies DVFS knob to downscale the voltage and frequency of applications that are currently running on the chip. For voltage down/up scaling (in Algorithm 1 lines 3-4, 12-14), we use the approach presented in [19]. If power violation persists, the PG knob is used to power gate all the cores that are currently un-occupied by any tasks on the chip. Although, this leads to increase in dark silicon, and further accumulation of the same happens when workload intensity is higher. In case of AWT exceeding the threshold, APPX knob is invoked by switching the mode to approximate execution. A decision to map any incoming applications in their approximate mode is indicated to the RMSU. The applications currently running on the chip that are approximable are switched to their approximate mode. A mapping command is sent to the RMSU indicating the application(s) to be switched and level of approximation. If AWT is still violated, applications that are already running in level-1 approximation are further switched to level-2 of approximation. The level of approximation can be set in a fine-grained manner by using N-levels of accuracy trade-offs. For instance, Palomino *et. al* have used a 4-level approximation in video encoding to optimize chip's temperature [17]. In this work, we use two levels of approximation to demonstrate energy gains. If TDP is honored, voltage and frequency levels of applications that are currently down-scaled are cranked up. If the TDP is still honored after voltage upscaling, un-occupied cores that are power gated are powered up. While AWT is honored, mode of incoming applications is changed back to accurate. If AWT is honored further, a mapping command indicating the application and new level of approximation (i.e., more accurate level) is sent to the RMSU. Mapping and mode switching decisions made by RMSU are presented in Algorithm 2. The RMSU receives mode of execution for incoming applications from the power manager. Upon arrival of a new application, the RMSU buffers approximate tasks of the application (if any) into the Task Buffer. By default, RMSU maps accurate version of an application on to the chip if the *inMode* is ACC, while the approximate mode is mapped when *inMode* is APPX. RMSU receives mode switching commands for current set of applications running on the chip. The *runMode* command indicates the application whose task has to be replaced and the level of approximation. When *runMode* of a specified application is set to level-1, the RMSU fetches corresponding approximate task (T') from the Task Buffer and replaces the accurate task with approximate task. In case of *runMode* of an application being set to level-2, RMSU repeats the aforementioned switching process, with an approximate task of level-2. When the power and workload intensity are manageable within their thresholds, we switch back to ACC mode

Algorithm 2 Mode Switching Algorithm

Inputs: $inMode$: Execution mode for incoming applications, $runMode$: Execution mode of currently running applications; App_{in}: Incoming applications,App'_{in}: Approximate version of incoming application, App_R: Application currently running;
Outputs: Map: Mapping configuration of incoming application;
Constants: TDP: Power budget, AWT_{th} : AWT Threshold

Body:

1: **if** newApp **then**
2: $taskBuffer.push(App.T', App.T'')$;
3: **if** $inMode = APPX$ **then**
4: $Map(App'_{in})$;
5: **else**
6: $Map(App_{in})$;
7: **for** App_R **do**
8: **if** App_R.runMode = level-1 **then**
9: wait until current iteration of T finishes;
10: switch(T,taskBuffer(T'));
11: **else if** App_R.runMode = level-2 **then**
12: wait until T' finishes;
13: switch(T',taskBuffer(T''));

by indicating the same to the RMSU. Inspired by the technique presented in [18], we avoid oscillating between ACC and APPX by setting intermediary thresholds TDP_{th} and AWT_{th}. We switch from ACC to APPX mode when the power and workload are approaching their thresholds, while switching back to ACC mode is invoked only when these parameters approach intermediary thresholds. This ensures that switching back to ACC mode is done when power and workloads are maintained well under their thresholds and minimizes the chance of oscillating between the two modes frequently. Setting intermediary thresholds is a designer's decision, with a choice between quality and throughput.

5. EVALUATION

We assess the efficiency of our approximation-enabled power management approach, APPEND, against state-of-the-art dynamic power management/capping techniques PG [7] (based on PCPG) and and MOC [19] (based on per-core DVFS and PG). We chose widely used data-triggered on-line learning applications that fall under classification and estimation. They are inter-disciplinary, being used in a range of periodic applications like recognition, mining and automation that are performance and energy demanding. These workloads are based on iterative methods of computation, meaning that the accuracy of result converges towards optimal solution with more number of iterations. Since an accurate solution may not exist and lower convergence could still offer an acceptable result, they become candidates for approximation. The list of applications used for evaluation is presented in Table 1. We normalize the performance gain of approximate tasks of level-1 (AP-1) and level-2 (AP-2) in comparison with their accurate versions. For level-1 and level-2 of approximation with loop perforation (LP) for linear regression and least squares, we skip 10% and 25% of computations on input data respectively. For k-means clustering and k-nearest neighbors, we use relaxed convergence (RC). We compromise on number of flips, coverage of neighbors and training data sets respectively for these applications. We set the limits of relaxation on convergence to 1% and 5% respectively for two levels of approximation. For each application and level of approximation, we present normalized energy gain when compared to accurate tasks. It can observed that performance and energy gain increases with amount of accuracy traded. These gains come at the loss of accuracy, ranging from 3% to 13% over different applications and data sets. Accuracy trade-offs and overheads incurred in switching mode of execution from accurate to approximate are reasoned in Section 5.2.1.

5.1 Simulation Environment

Applications are modeled as task graphs, as described in Section 4. We implement each application on interval-core based Sniper simulator, annexed with McPAT for modeling power [23]. We used Nehalem-like processing elements with 32KB of instruction and data caches. We model each application as a combination of concurrent tasks, preserving data flow nature. We use loop perforation and relaxed convergence in case of approximate tasks. We extract execution time, average power and energy consumption per each task. We normalize these values as *compute factor* metric for each node in the task graph, along with amount of data flow as the *communication volume* between tasks. The task graph for each application is thus a directed network of nodes that holds execution time, communication volume, average static and dynamic power consumption for accurate and approximate tasks. We use our in-house cycle accurate simulator implemented in SystemC to evaluate the proposed power management framework. We extended Noxim [6] NoC simulator using its network infrastructure for interconnects.

Table 1: Applications' Energy-Accuracy Trade-offs

App	Appx	Norm. Perf		Norm. Energy		% Error	
		AP-1	AP-2	AP-1	AP-2	AP-1	AP-2
Linear Regression	LP	1.1	1.2	1.16	1.27	6	13
K-Means	RC	1.93	5.6	1.9	5.27	1	5
K-NN	RC	1.11	1.33	1.09	1.22	3	8
Least Squares	LP	1.07	1.26	1.11	1.25	5	12

The power characteristics of processing elements (PE) are modeled based on metrics extracted from McPAT and Lumos [25]. Lumos is an analytical framework that quantifies power-performance characteristics with technology node scaling for many-core systems. We used Lumos for physical scaling parameters, voltage scaling and TDP metric for different network sizes. We added the support for dynamic arrival and servicing of applications through the run-time mapping unit. The mapping unit receives commands from power controller, implemented as a software module. The test-bed is a rectangular network with X-Y routing. The $tile_{(0,0)}$ of the mesh acts as the central manager that is responsible for keeping track of mapping information. The network size is 12×12 and the chip area is $138mm^2$. For the *first node* selection in the runtime mapping process, we use MapPro [8] method. For the DVFS purpose, we use 15 VF levels with voltage in the range of 0.8V-1.2V. The frequency of the on-chip communication network (e.g., routers) is set to the maximum level (similar to [7] and [19]). The TDP value is set to 90W, based on the chip's power density.

5.2 Evaluation Metrics and Results

For evaluation purposes, we simulate the system over a period in which 200 applications are serviced. The evaluation metrics are: i) *Power Consumption*: Power consumption of the system over the period of execution, honoring TDP by capping the power, ii) *Accumulated Wait-time*: Accumulated value of wait-time of applications before the application request is serviced, and iii) *Throughput*: Time consumed to service 200 applications Our pre-requisite objective is to cap the power consumption such that TDP constraint is honored throughout the period of execution. Figure 8 shows the power consumption of DVFS, PG, MOC and APPEND, along with TDP constraint, over the execution time for servicing 200 applications. TDP violation is more frequent with PG and DVFS knobs, while TDP is honored for most of the execution period with MOC and APPEND, with APPEND being better of the two. Noticeable issue is that while honoring TDP, APPEND maintains power consumption closest to TDP when compared to other knob combinations, reflecting better utilization of available power budget. This indicates mitigation of dark silicon and can be attributed to hierarchical usage of power knobs in APPEND's power controller. Moreover, we actuate power knobs - DVFS and PG by monitoring power consumption over an epoch e_1 and trigger the approximation knob pro-actively over epoch e_2 with e_2 being five times longer than e_1. This eliminates possible random actuations or oscillations between different modes of execution. With better utilization, APPEND is able to service applications faster, reducing the run-time and consequently wait-time of incoming applications. Figure 9 shows the accumulated wait-time (AWT) for different power capping actuators over the period of execution. We present AWT as a function that is directly related to rate of application requests made. Similar to power capping, APPEND has the best AWT, preceded by MOC, PG and DVFS. DVFS and PG based actuations have higher AWTs already when the application request rate reaches 3 per second. MOC has a relatively high AWT when application request rate is 5 per second. However, APPEND has a near-zero AWT for as long as 5× more than DVFS and PG and 3× more than that of MOC. This demonstrates the ability of APPEND

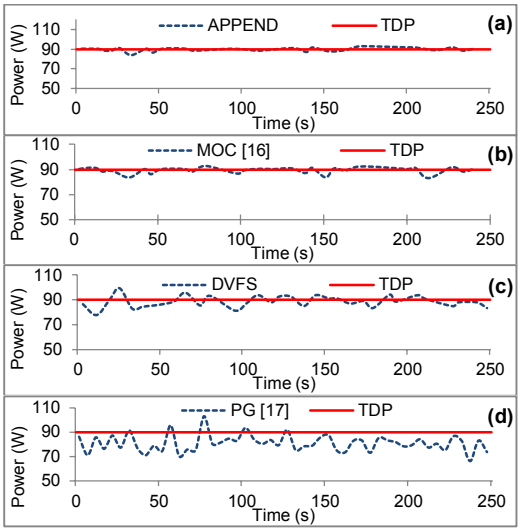

Figure 8: Instantaneous Power Consumption

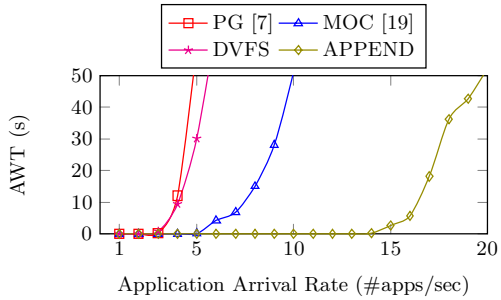

Figure 9: Accumulated waiting time

to service applications faster despite high workloads, when compared to the other knobs. APPEND has *AWT* greater than zero when the application request rate reaches 14 per second. Also, *AWT* accumulation is more steeper in case of other knobs than that of APPEND, indicating a substantial rise in their wait-times with high request rates. The minimal *AWT* and high service rate of APPEND also results in high throughput and energy efficiency. Normalized gain in throughput for all knob combinations is shown in Figure 10. APPEND has a throughput that is 1.5× better than PG and 1.2× better than MOC, showing a significant gain in performance and energy while power capping is strictly maintained. Employing APPX knob allows APPEND to minimize execution time of applications running on the chip. With applications leaving the system faster, more resources (cores) become available for incoming applications and reduces their wait-time. APPEND benefits from AWT and throughput mutually improving each other.

5.2.1 Error and Overhead Analysis

Behavioral patterns of accurate (Acc) and approximate (Appx) versions of each application are shown in Table 2. The number of instructions of each applications (Instructions), number of instructions simulated (Instr. Sim (M) in million), number of L1-instruction cache accesses (L1-I Accesses (M)) (in million), and normalized overhead (in %) are presented in Table 2. Number of instructions are slightly higher for approximate tasks due to conditional branching involved. However, these instructions eventually result in reduced overall workload and hence improve performance. For each application, we used 1 million elements in training set in increasing steps of 100000 data points per period. Number of simulated instructions depend on training and test data sets used, and are variable in case of different sizes of data used. With loop perforation and relaxed convergence, input data elements are skipped, resulting in fewer instructions required to be simulated. Normalized energy savings, performance gain and loss in accuracy for each application are shown in Table 1. For loop perforated (LP) applications, we used squared distance from accurate solution to calculate the error and for relaxed convergence (RC), we set 1% and 5% as limits for convergence. Switching execution from accurate to approximate version incurs some overhead due to monitoring and triggering the approximate version. For every approximate task, the switching of execution mode involves a conditional branching instruction(s). The overhead incurred during this transformation included in the approximate task's *compute factor*. For the applications we used, the normalized overhead penalty incurred ranged between 0.3% up to 1.2%. This overhead is negligible when compared to the workload reduced by approximation and thus levies no significant performance penalty. Further, loading an approximate task involves moving new instruction stream to the instruction cache, with a possibility of increase in the number of DRAM accesses. However, this depends on the number of application instructions and the size of L1-instruction cache. For instance, a larger application coupled with smaller L1-instruction cache presents a worst case scenario that would force the system to evict accurate task and fetch the approximate task from main memory. Although, in our testbed, we used L1-I cache of 32KB and all the applications have instructions up to as many as 1500. The worst case penalty in terms of communication for switching from accurate version of a task to the approximate version can be calculated as follows:

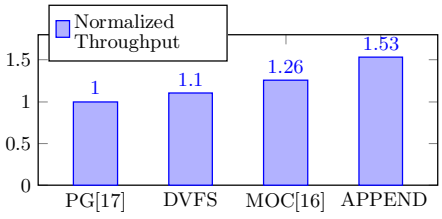

Figure 10: Normalized throughput

Table 2: Applications - Behavior and Overhead

App	Instr.		Instr. Sim (M)		L1-I Accesses (M)		Overhead % (Norm.)
	Acc	Appx	Acc	Appx	Acc	Appx	
Linear Regression	1105	1150	93	75	9	7	1.2
K-Means	1018	1021	449	102	57	14	0.3
K-NN	1457	1513	140	105	37	24.7	0.3
Least Squares	770	815	53	50	5	3.7	1.2

$$penalty = \frac{size_{app}}{size_{pkt}} \times (P_L + (n \times r_L) + MC_L + DRAM_L) \quad (2)$$

where $size_{app}$ and $size_{pkt}$ are application and packet sizes, P_L is packetizing latency i.e., time consumed to packetize data, access the network interface and inject packets into the network, n is the number of hops from a core to nearest memory controller, r_L is router channel latency, MC_L and $DRAM_L$ are access latencies of memory controller and off-chip memory. We demonstrate worst case overhead penalty of mode switching for a video encoding application which was used by Holmbacka *et al.* as an example that they used to demonstrate overhead for task migration [10]. For experimental many-core platform Intel SCC, the core, network and off-chip memory frequencies are 533MHz, 800MHz and 400MHz, respectively. The worst case mode switching penalty using SCC for the video encoding application of size 6KB is 1.5ms. For the same application, penalty in task migration is 10.6ms, 7× more than the mode switching over-

head, to move both instructions of 6KB and data of 16KB. Task migration overhead can still be higher when more data is to be moved, while mode switching needs no movement of data. It should be noted that these values are subjective to the platform on which they are executed, while the relative difference in overheads between mode switching and task migration might hold good.

6. RELATED WORK

Power Capping: Adaptive Power capping techniques monitor the power consumption and actuate power knobs in a closed loop, in case of TDP violation. A PID controller based power management is presented in [7], where knob settings are actuated for power capping as per normalized gain of PID. Vega et. al propose a power capping algorithm using DVFS, PCPG and core folding, with all power knobs tightly coupled [24]. They suggest that combinatorial usage of different power knobs is effective for system level power capping decisions. Cochran *et al.* have used thread packing i.e., allocation of threads per core as a power knob along with adaptive DVFS [4]. PGCapping was presented in [13] that uses PCPG and DVFS in a hierarchical way for power capping and life time balancing. Kapadia *et al.* have used Degree-of-parallelism (DoP) as a knob for power management and to improve system reliability. Application mapping i.e., spatial alignment of active cores for improving power budget and thus power capping limit was proposed in [11] and [21]. A multi-objective power capping approach was presented in [19] which uses combination of DVFS and PCPG based on network and workload characteristics. Chen et. al have proposed using resource allocation at data center level as another knob for power actuation [3]. They use history based prediction for potential workload to determine CPU resource allocation. While all the above techniques use TDP as upper bound, Pagani et. al have proposed an adaptive way way of setting the upper bound on power consumption, thermal safe power (TSP), as a function of spatial alignment of active components. [16].

Approximation: Ansel *et al.* have used variable accuracy implementations of same algorithm, with language and compiler support to choose one among different implementations for exploring energy-accuracy trade-offs. [1]. Baek and Chilimbi have proposed approximation at software level with a choice between accurate and approximate versions of blocks of code using Green compiler [2]. Hoffman *et al.* have proposed using energy-accuracy trade-offs in context of power capping by translating static parameters of an application into dynamic knobs such as convergence for drop in accuracy [9]. However, other approximations at algorithmic level such as logic simplification cannot be translated into dynamic knobs. Escaping infinite loops and skipping iterations of long bottleneck loops was proposed by Sidiriglou *et al.* as Loop Perforation [22]. All these techniques explore ways to compute approximately, keeping quality control, energy and performance gains in view. However, they do not use approximation for actuating power consumption in a closed-loop way.

7. CONCLUSIONS AND FUTURE WORK

In this work, we proposed approximation as another knob for power capping in many-core systems. We used the APPX knob hierarchically with other power knobs of DVFS and PG to gain performance and energy within the power budget, for some accuracy trade-offs. We presented a power management framework, APPEND, that monitors power consumption and workload intensities to dynamically replace accurate tasks with approximate tasks. We used multiple variable accuracy implementations of the same application to support dynamic switching between tasks. APPEND thoroughly honors TDP and yields higher throughput and lower wait-times, in comparison with state-of-the-art power management techniques. Our proposed approach requires application design support and restricts the accuracy trade-offs to a fixed scale. Making the system aware of possibilities in accuracy scaling for a fine-grained and dynamic approximation is planned for future work.

Acknowledgment

The authors acknowledge financial support by Academy of Finland project, "MANAGE: Data Management of 3D Systems for the Dark Silicon Age" and University of Turku graduate school (UTUGS).

8. REFERENCES

[1] J. Ansel et al. PetaBricks: a language and compiler for algorithmic choice. *ACM SIGPLAN Notices*, 2009.
[2] W. Baek et al. Green : A Framework for Supporting Energy-Conscious Programming using Controlled Approximation. In *PLDI*, 2010.
[3] H. Chen et al. Dynamic server power capping for enabling data center participation in power markets. In *ICCAD*, 2013.
[4] R. Cochran et al. Pack & cap: adaptive dvfs and thread packing under power caps. In *MICRO*, 2011.
[5] H. Esmaeilzadeh et al. Dark silicon and the end of multicore scaling. In *ISCA*, 2011.
[6] F. Fazzino et al. Noxim: Network-on-chip simulator. *URL: http://sourceforge.net/projects/noxim*, 2008.
[7] M. Haghbayan et al. Dark Silicon Aware Power Management for Manycore Systems under Dynamic Workloads. In *ICCD*, 2014.
[8] M. Haghbayan et al. MapPro: Proactive Runtime Mapping for Dynamic Workloads by Quantifying Ripple Effect of Applications on NoCs. In *NOCS*, 2015.
[9] H. Hoffmann et al. Dynamic knobs for responsive power-aware computing. *ACM SIGPLAN Notices*, 2012.
[10] S. Holmbacka et al. A task migration mechanism for distributed many-core operating systems. *Journal of Supercomputing*, 68(3), 2014.
[11] A. Kanduri et al. Dark silicon aware runtime mapping for many-core systems: A patterning approach. In *ICCD*, 2015.
[12] N. Kapadia et al. VARSHA: Variation and Reliability-aware Application Scheduling with Adaptive Parallelism in the Dark-silicon Era. In *DATE*, 2015.
[13] K. Ma and X. Wang. PGCapping: Exploiting power gating for power capping and core lifetime balancing in CMPs. *PACT*, 2012.
[14] S. Misailovic et al. Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels. In *OOPSLA*, 2014.
[15] T. Muthukaruppan et al. Hierarchical power management for asymmetric multi-core in dark silicon era. In *DAC*, 2013.
[16] S. Pagani et al. TSP: Thermal Safe Power: Efficient Power Budgeting for many-core systems in dark silicon era. In *CODES+ISSS*, 2014.
[17] D. Palomino et al. Thermal optimization using adaptive approximate computing for video coding. *DATE*, 2016.
[18] A. Rahmani et al. Design and management of high-performance, reliable and thermal-aware 3D networks-on-chip. *IET Circ., Dev. & Sys.*, 2012.
[19] A. Rahmani et al. Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach. In *ISLPED*, 2015.
[20] A. Rahmani et al. *The Dark Side of Silicon*. 2016.
[21] M. Shafique et al. Dark Silicon As a Challenge for Hardware/Software Co-design. In *CODES+ISSS*, 2014.
[22] S. Sidiroglou et al. Managing performance vs. accuracy trade-offs with loop perforation. In *FSE*, 2011.
[23] E. Trevor et al. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations. In *SC*, 2011.
[24] A. Vega et al. Crank it up or dial it down: Coordinated multiprocessor frequency and folding control. In *MICRO*, 2013.
[25] L. Wang and K. Skadron. Dark vs. dim silicon and near-threshold computing extended results. *Univ. of Virginia, Dept of Comp.Sci Technical Report*, 1, 2012.