Accuracy Aware Power Management for Many-core Systems running Error Resilient Applications

Anil Kanduri, *Student Member, IEEE,* Mohammad-Hashem Haghbayan, *Student Member, IEEE,* Amir M. Rahmani, *Member, IEEE,* Pasi Liljeberg, *Member, IEEE,* Axel Jantsch, *Member, IEEE,* Hannu Tenhunen, *Member, IEEE,* Nikil Dutt, *Fellow, IEEE*

***Abstract*—Power capping techniques based on dynamic voltage and frequency scaling (DVFS) and power gating (PG) are oriented towards power actuation, compromising on performance and energy. Inherent error resilience of emerging application domains such as Internet-of-Things (IoT) and machine learning provide opportunities for energy and performance gains. Leveraging accuracy-performance trade-offs in such applications, we propose approximation (APPX) as another knob for close-looped power management, to complement power knobs with performance and energy gains. We design a power management framework, APPEND+, that can switch between accurate and approximate modes of execution subject to system throughput requirements. APPEND+ considers the sensitivity of the application to error to make disciplined alteration between levels of approximation such that performance is maximized while error is minimized. We implement a power management scheme that uses APPX, DVFS and PG knobs hierarchically. We evaluated our proposed approach over machine learning and signal processing applications along with two case studies on IoT - early warning score system and fall detection. APPEND+ yields 1.9× higher throughput, improved latency up to 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*—Approximate Computing, Power Management, Dark Silicon; Runtime Mapping; Internet-of-Things**

I. Introduction

EMERGING application domains such as Internet-of-things (IoT), cyber-physical systems (CPS) and big data analytics etc., are compute intensive and power hungry [1]. Transistor scaling supported building denser chips that provide higher compute intensity to meet performance requirements of these applications, while keeping the power density constant. With transistor scaling reaching its physical limit, operating voltage approaches its threshold and cannot be further scaled down gracefully with transistor scaling [2]. This leads to rise in power density and subsequently thermal violation. Performance surges of emerging application domains, smaller chip areas and limited cooling solutions contribute to high power densities and frequent thermal violations, potentially damaging the chip's functionality. To avoid thermal violations, the chip has to function within dissipatble (safe) limits of power. This forces a section of chip to be powered off temporally - this inactive portion is termed as dark silicon [3]. Dark silicon phenomena reduces performance, energy efficiency and utilization of on-chip resources [2].

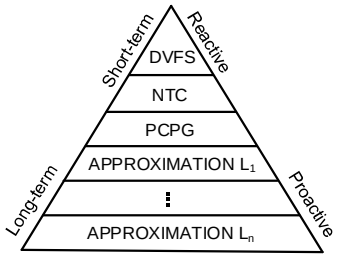


Figure 1: Power Management Knobs

Power capping techniques are used to restrict power consumption of the chip to a fixed and safer limit, typically a design time estimate called thermal design power (TDP), beyond which thermal violations may occur [4]. Dynamic power capping and management techniques typically function in an observe-decide-act loop - observe instantaneous power consumption and temperature accumulation, decide on power actuation, and act on the decisions through power knobs [5]. Dynamic voltage and frequency scaling (DVFS), power gating (PG) [6], near threshold computing (NTC) [7] and adaptive scheduling [8] are widely used knobs for power management. Combinatorial actuation of different power knobs can honor thermal and power constraints, although performance and/or energy gains can be minimal [9]. DVFS knob would be limited as the voltage approaches its threshold and cannot be scaled down any further and also suffers with increase in leakage power. Power Gating (PG) knob addresses the issue of static power and is not limited as DVFS. However, the reduction in static power comes at the expense of performance, since only fewer cores are simultaneously powered up. Both DVFS and PG are triggered as a reaction to power violations, which might work for instantaneous power reduction in short-term. Despite power capping benefits, they do not offer any substantial gains on performance or energy efficiency.

Approximate computing is emerging as an alternative for dark silicon mitigation, by trading off accuracy for performance and energy gains. Applications from several domains are inherently error resilient, based on their nature of computation and/or input data. For example, algorithms used in multi-media signal processing, machine learning and numerical methods etc., can be iterative or NP-hard, making them tolerant to inaccurate computations. Similarly, IoT systems

Anil Kanduri, Mohammad-Hashem Haghbayan, Pasi Liljeberg, and Hannu Tenhunen are with University of Turku, 20500 Turku, Finland. Axel Jantsch is with TU Wien, 1040 Vienna, Austria. Amir M. Rahmani and Nikil Dutt are with University of California, Irvine, CA 92617 USA.
E-mail: spakan@utu.fi, mohhag@utu.fi, amirr1@uci.edu, pakrli@utu.fi, axel.jantsch@tuwien.ac.at, hannu@kth.se, dutt@uci.edu
Hannu Tenhunen is also with Royal Institute of Technology (KTH), Kista, Sweden, 16440.
Amir M. Rahmani is also with TU Wien, 1040 Vienna, Austria.

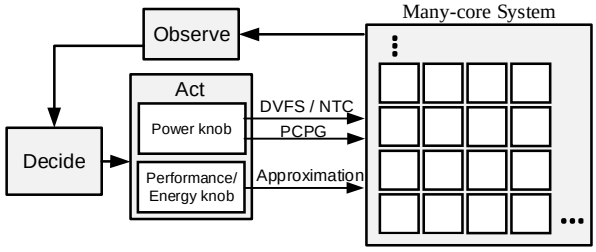


Figure 2: The Approximation Knob

deal with continuous sensory data originating from analog or noisy sources, where relaxing certain computations has less effect on eventual result. Approximation leverages inherent error resilience of such applications to reduce computational workload and increase energy efficiency.

Alongside soft computing applications, several IoT applications rely on a smart gateway for edge and/or cloud based processing [10]. With several sensor nodes mapped to a single edge or cloud processor, performance and power challenges continue to effect IoT applications indirectly. Traditional power knobs alone would not suffice to ensure power capping while maintaining higher performance and energy efficiency. To fill the performance and energy gap of power knobs, we propose approximation (APPX) as another knob for power capping and management. We couple the short-term reactive power knobs - DVFS and PG, with the long-term pro-active energy and performance knob - APPX, in a hierarchical manner for power capping, while ensuring performance and energy gains, at the expense of accuracy. Figure 1 shows power knobs classified in the order of their temporal effect and overhead.

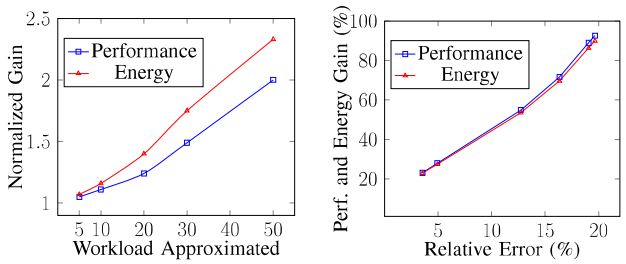
In this paper, which is major extension of our recent work published in [11], we propose an approximation enabled power management framework APPEND+, that uses DVFS and PG knobs primarily for power capping and APPX knob for performance. Figure 2 shows the top-level view of approximation as another knob for power management along with conventional power knobs. We design a power manager that makes decisions on actuation of DVFS and PG knobs in case of power violation and APPX knob in case of throughput violation. The key idea of this work is to switch the mode of execution of an application from accurate to approximate upon performance requirements with APPX knob invocation. In our previous work, we switch from accurate to approximate mode of execution among approximable applications, subject to system requirements [11]. At times, this strategy either over compensates for performance surges by approximating beyond the requirement, or falls short of meeting the performance requirement. We fill this gap using *sensitivity metric* for each application at each level of approximation that is used to make mode switching decisions. We use a set of variable accuracy implementations of an aprpoximable task, with each approximate task identified by its sensitivity metric which is represented as the performance gained per error induced. To enable selection of suitable candidates for mode switching, we prune this set to choose a candidate task for replacing the accurate task that offers maximal performance gain within minimal error. We present a run-time mapping and mode switching algorithm for replacing accurate tasks with approximate tasks from the set of variable accuracy implementations. Our contributions based on our prior work [11] are as follows.

- Approximation knob for closed loop power and performance management
- A classification algorithm for identifying approximable tasks that maximizes performance by pruning application space based on sensitivity metric
- A run-time mapping and mode switching technique for replacing accurate tasks with approximate tasks
- A power management framework APPEND+, that uses DVFS, PG and APPX hierarchically for power capping and throughput improvement
- A case study of IoT applications - fall detection and early warning score (EWS), to evaluate APPEND+

II. Related Work

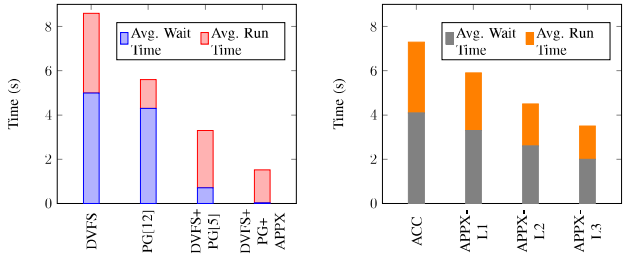
Power Capping: Power capping techniques monitor the power consumption and actuate power knobs in a closed loop, in case of power consumption exceeding TDP. A PID controller based power capping is presented in [12], where knob settings are actuated as per normalized gain of PID. Vega *et. al* [13] propose a power capping algorithm using DVFS, PCPG and core folding, with all power knobs tightly coupled. They suggest that combinatorial usage of different power knobs is effective for system level power capping decisions. Cochran *et al.* [14] have used thread packing, i.e., allocation of threads per core as a power knob along with adaptive DVFS. PGCapping was presented in [6] 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 [8]. Application mapping i.e., spatial alignment of active cores for improving power budget and thus power capping limit is proposed in [15] and [16]. A multi-objective power capping approach is presented in [5], [17] which uses combination of DVFS and Per-core Power Gating (PCPG) based on network and workload characteristics. Chen *et. al* [18] have proposed using resource allocation at data center level as another knob for power actuation. 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* [4] 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. All these techniques focus exclusively on power capping and combinatorial usage of power knobs, but do not consider their implications on performance.

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 [19]. Baek and Chilimbi have proposed approximation at software level with a choice between accurate and approximate versions of blocks of code using Green compiler [20]. Hoffman *et al.* [21] 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. However, other approximations at algorithmic level such as logic simplification cannot be translated into dynamic knobs. Escaping infinite loops and skipping iterations of bottleneck loops that consume longer execution time was proposed by



(a) Workload-Performance trade-offs for matrix multiplication (b) Accuracy-Performance trade-offs for k-means clustering

Figure 3: Performance and Energy gains with approximation



(a) Application Service Time for different Knobs (b) Application Service Time for different levels of APPX Knob

Figure 4: Performance gains of different knobs for k-means clustering simulated on 16-core system

Sidiriglou *et al.* as Loop Perforation [22]. All these techniques explore ways to compute approximately for energy and performance gains, within acceptable quality. However, they do not use approximation for closed-loop power actuation.

IoT: In the context of IoT applications, the need for computational capacity at a sensor node level would not suffice. To meet real time performance requirements, data collected over sensory nodes is processed at a smart gateway [10]. The gate way acts an intermediate edge layer between the sensor front end and the cloud server back end [23]. Typical gate way can be a multi-core platform, which still faces with power and energy consumption challenges [24]. Some of the IoT applications are concerned with actuation mechanism based on sensory data analysis such as identifying sudden changes in input data. Such applications present with an opportunity to relax the accuracy of computation which in turn can be used to conserve energy and accelerate the performance.

III. The Approximation Knob

Approximate execution of an application provides variable performance and energy gains with the amount of error induced. We present the pareto-space of accuracy-performance trade-offs with two example applications viz., sparse matrix multiplication and k-means clustering. We considered two 10000 $\times$ 10000 sparse matrices, that are multiplied approximately by skipping inner most loop that performs multiplied accumulation. Figure 3(a) shows normalized performance and energy gains with the number of inner most loops that are skipped (workload reduced) to reduce number of computations. With 50% of workload reduced, performance and energy efficiency doubles. For k-means clustering, we use 10000 random input data points to be classified into 50 clusters. We used relaxed convergence as the approximation, by early termination of the clustering algorithm with non-zero flips. Figure 3(b) shows the gain in performance and energy for error induced by relaxing the convergence from 5%-10%. Relative performance gains increase as the error induced increases. Both these examples establish performance and energy gains with approximation. Specifically, they provide an insight on performance gains per error induced, which can be exploited while implementing the APPX knob.

We demonstrate the impact of different power knobs on performance of many-core systems with applications dynamically entering and leaving the system using k-means clustering as an example. The performance of the system 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 [8]. Dynamic workload characteristics contribute to power violations, forcing actuation of power knobs. We simulate the application for 4 different power knobs viz., DVFS, PG, DVFS+PG (referred as MOC) [5], and DVFS+PCPG+APPX, using the experimental platform, detailed in Section VII. The APPX knob has k levels of approximation, where k is a parametrizable entity. In this case, we chose 4 levels of approximation.

The average service time for different knob combinations is shown in Figure 4(a). In case of using DVFS and PG knobs [12], per-application run-time increases forcing 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 [5], [17]. 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. Figure 4(b) shows application service time of APPX knob over different levels of approximation. The gain in performance with increasing level of approximation is trivial. 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.

IV. Accuracy-Performance Trade-offs: A Case for IoT

IoT applications deal with real world sensor data that is analog and involves noisy components. Performance requirements for IoT systems are usually high while energy budget is limited [1]. Collection and classification of raw data into meaningful clusters, filtering, computation for actuation decisions, and communicating external world are major functionalities of an IoT system. Every stage in this process has a variable tolerance to inaccurate computations, presenting opportunities for approximation. Leveraging this fine-grained error resilience, approximation can provide better performance-per-energy in IoT systems. We explore the possibilities of accuracy-performance trade-offs in IoT domain using two case studies on health monitoring applications viz., early warning score (EWS) system and fall detection.

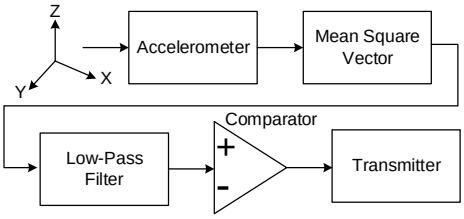


Figure 5: Fall Detection

A. EWS System

An Early Warning Score (EWS) system is used in health care for monitoring vital signs of a patient to proactively alert medical support. A method for EWS is proposed in [25] that uses three types of sensors - medical, environmental and activity. Data collected from the senors is pre-processed by using a Butterworth filter to remove noise components and false alarms. Sensory data from different sources is fused to extract useful details. Every physiological data sensed is allocated a score based on the range the sample belongs to and its implication on patient's health deterioration. The final score is calculated as a combination of scores of all the individual sensor nodes' data. When the final EWS exceeds a fixed threshold, a warning signal is transmitted seeking for medical attention. This system deals with heterogeneous data generated by different sensors and involves computations on unclassified, redundant and noisy data. To extract meaningful insights, the EWS uses data acquisition, filtering, fusion, classification, and analysis, followed by computation and transmission. Although this system is mission critical, there are several intermediary stages where inaccurate computations can be tolerated. For example, heart rate measured is classified into one of several ranges of heart rate data and a score is assigned according to the range it belongs to, but the exact value of heart rate is not used. Medical sensors trace several samples of data per second on an average, while a median of this data is good enough to represent all the samples. Relaxing such computations and data points that do not affect final EWS can enhance performance of the system within a lower energy budget.

B. Fall Detection

Another example of IoT application that is data and compute intensive is fall detection [26]. Fall detection mechanism identifies whether a person using the wearable detector falls hazardously on the ground. Specifically, fall detection is used in the context of patient and elderly people monitoring, to bring attention and support upon a fall. Typical fall detection employs camera, gyroscopic or accelerometer sensors for identifying a fall with respect to inertial position. We use fall detection based on accelerometer, as proposed in [26]. The accelerometer data in three dimensions is used to calculate signal magnitude vector as the square root of sum of squares of signal component in each axis. This is fed to a low pass filter to generate discrete signal of positioning. Unusual spikes in the filtered data when compared to a fixed threshold represents the possibility of a fall, which is then transmitted to support system infrastructure for further assistance. The fall detection mechanism is shown in Figure 5. A major conundrum in fall detection is in identifying the abnormal spikes in positioning signal - whether to analyze the accelerometer data tightly coupled to the sensor or to transmit the data to a cloud computer. Analyzing the data in a simpler micro-controller has performance penalties while transmitting filtered data to a high-end cloud computer consumes more energy. Expensive floating point computations on high sampled accelerator data such as multiplications, square root and filtering can be relaxed to gain performance. We have run the fall detector mechanism over 3 sample persons for 8 hours of a day. Sensory data is collected at 100 samples/sec and fall detection is executed at a gateway between sensor nodes and cloud server. These test cases show that accelerometer signals generate data that is usually redundant, indicating that skipping some of the sensory data samples would induce only a tolerable error. Reducing the sampling of sensor and filter length by half produced results that are similar to accurate computations. Relaxing accuracy of data analysis can thus improve fall detection's performance.

V. Knob Actuation Scenarios

We primarily monitor power consumption, workload intensity, and sensitivity of applications to make knob actuation decisions. The threshold for power consumption is TDP. Power consumption exceeding TDP indicates a power violation. Further, we also set another parameterizable threshold TDP_{th}, a metric that indicates possibility of a potential TDP violation, such that $0.66 \times TDP < TDP_{th} < TDP$. 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 given as:

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

Longer application wait time indicates higher the workload intensity. A parametrized metric AWT_{th} is set as threshold for workload intensity. The power-objective is to restrict the power consumption to TDP and throughput-objective is to restrict the AWT to AWT_{th}. For actuation of APPX knob, we use sensitivity metric of an application to choose an application and its corresponding level of approximation.

Application Sensitivity: The performance and energy gains varies for different applications over different levels of approximation. This presents a case where approximating one application might yield more performance gain than that of the others. We identify each application with a *sensitivity metric* as performance that could be gained per error induced. The motivation behind this is to choose an application that results in higher performance gain for the amount of error induced.

We define an application's sensitivity metric as:

$$Sensitivity = \frac{Perf_i - Perf_{i-1}}{Error_i - Error_{i-1}} \quad (2)$$

$Perf_i$ and $Error_i$ represents performance and error induced at i^{th} level of approximation. Sensitivity of an application for any two given levels of accuracy would be high when the performance gained by lowering accuracy is high or when the accuracy loss in performance improvement is lower. Subject to application characteristics and input data, sensitivity of an application varies through different levels of approximation. Sensitivity metric of an application presents a wider pareto-space of accuracy-performance trade-offs that can be explored

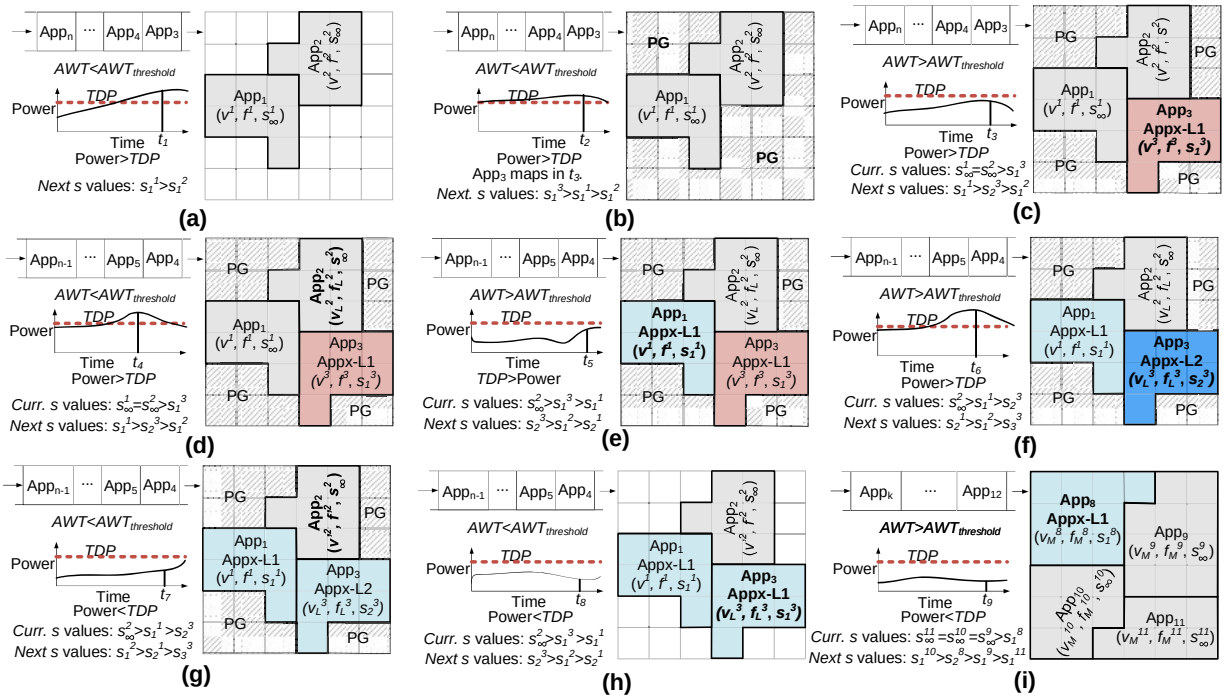


Figure 6: Knob Actuation Scenarios. Each scenario shows applications running on a 36-core system along with power consumption and workload intensities.

in choosing an application to be approximated and the level of approximation. Sensitivity metric identifies tasks that will result in highest performance gain among the set of tasks currently running, and prioritizes these tasks as candidates for switching their mode of execution to approximate. This enables fine-grained control on APPX knob, appropriate choices for mode switching, performance and energy gain at lower relative error and limits the possibility of over-compensation with approximation. Details on sensitivity metric used for evaluation purpose are detailed in Section VII. We demonstrate possible scenarios that require knob(s) actuation under diverse power consumption and workload intensities. Figure 6 summarizes these scenarios, representing power consumption, workload intensity and knob actuations employed over a span of execution. Each scenario (a-i) 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 and application's sensitivity. Voltage and frequency levels of each mapped application (App_1, App_2, ..., App_n) are represented as ((v^1,f^1,s^1), (v^2,f^2, s^2), ..., (v^n,f^n, s^n)). The lowest and highest levels of voltage and frequency are (v_L,f_L) (v_M,f_M) respectively. The corresponding lowest and highest levels of sensitivities are represented as s^1_∞ (lowest level is s_∞ as error is 0) and s^1_m. The sensitivity for each application at different levels of approximation is shown as s^n_i, where n is the application number and i is the level of approximation. Criteria for knob actuation are TDP violation (power > TDP) and high request rate of incoming applications ($AWT > AWT_{th}$). **Scenario (a):** Two applications App_1 (v^1,f^1,$^1_\infty$) and App_2 (v^2,f^2,s^2_∞) are currently running in accurate mode of execution. At time t_1, a power violation ($power > TDP$) occurs, while the throughput is under control ($AWT < AWT_{th}$). DVFS knob is triggered over App_1 and App_2 to lower the power consumption within TDP. **Scenario (b):** App_1 and App_2 are now running at down-scaled voltage and frequency levels (v^1,f^1,s^1_∞) and (v^2_L,f^2_L,s^2_∞) with DVFS invocation at t_1. Although power consumption is relatively lower than at t_1, power violation still persists. So, PG knob is invoked to power gate the remaining un-occupied cores (power gated cores are shaded). This would potentially violate throughput constraint, since subsequently arriving applications do not have any free cores to be mapped onto. **Scenario (c):** Power consumption is below TDP and there is no power violation. Application request rate has increased and there is a throughput violation ($AWT > AWT_{th}$), due to power knobs triggered previously. APPX knob is invoked to counter throughput violation. App_3, arrived at t_3, has a higher sensitivity than App_1 and App_2, and hence is mapped in its approximate version (App_3(v^3,f^3,s^3_1)). Intuitively, we are reducing the execution time of App_3 so that there are free cores for potentially incoming applications. **Scenario (d):** At time t_4, the power consumption approaches TDP_{th}, indicating the possibility of power violation. DVFS knob is triggered to avoid potential power violation over App_2, which is chosen as candidate for voltage downscaling based on its network and compute characteristics (App_2(v^2_L,f^2_L,s^2_∞)). **Scenario (e):** At time t_4, power is under control, while throughput violation persists. App_3 is running in approximate mode at level-1. APPX knob is invoked again to address throughput violation, with a choice among switching App_2 to its next level i.e., level-2 of approximation, or App_1 to level-1 of approximation. The sensitivity metric for App_1 at level-1 s^1_1 is higher than the other two applications at level-2 (s^2_2, s^3_2), hence App_1 is chosen to switch to level-1 of approximation (App_1(v^1_L,f^1_L,s^1_1)). **Scenario (f):** At time t_5, power is under

control, while request rate is still high. All the approximable applications are running in approximate mode at level-1. APPX knob is invoked again, with a choice among the applications for maximum performance gain. The sensitivity metric for App_3 at level-2 of approximation is higher ($s_2^3 > s_2^2 > s_2^1$), and is thus chosen to switch the level of approximation further to level-2 (shown in bold) ($App_3(v_L^3,f_L^3,\mathbf{s_2^3})$). **Scenario (g):** At time t_5, power consumption is below TDP, so the DVFS knob is invoked to upscale the voltage and frequencies of some cores. Among the three applications, App_2 is chosen for up-scaling, as it benefits the most based on its network and compute characteristics. **Scenario (h):** At time t_8, power consumption is below TDP, and the application request rate is also below its threshold. DVFS knob is invoked to upscale voltage and frequencies of some more cores. Since App_2 is previously up-scaled, App_1 is now chosen as the candidate that benefits from up-scaling. Since the throughput constraint is maintained, APPX knob is invoked. App_3, which has a higher sensitivity is chosen to switch a level up in accuracy, going into level-1 of approximation from level-2 ($App_3(v_L^3,f_L^3,\mathbf{s_1^3})$). This invocation can be influenced by a user-defined parameter to up-scale voltage instead of switching up the level of approximation **Scenario (i):** At time t_9, power consumption is well below TDP, allowing more power to be consumed safely. DVFS knob is invoked to upscale the voltage and frequencies of all active cores to their maximum values. The application request is still higher than threshold, despite voltage up-scaling and hence APPX knob is invoked. App_8 has the highest sensitivity among running applications, hence it is chosen to switch mode of execution to approximate at level-1 ($App_8(v_M^8,f_M^8,\mathbf{s_1^8})$).

VI. System Design

We design our power management framework for NoC-based many-core systems supporting dynamic arrival of applications. Our power management framework monitors per-core power consumption and utilization, network intensity, incoming application request rate and sensitivities of applications to error to actuate different knobs accordingly. The top level view of our system architecture is shown in Figure 7.

A. Application Modeling

We model individual computational blocks of an application as a task. Each task is identified by its compute intensity, communication volume with other tasks, and power consumption. Applications are modeled as directed graphs, with each node representing a task running concurrently with other tasks. Analysis of applications' power-performance characteristics and task graph formation is implemented as an off-line function. Incoming applications are classified as approximable and non-approximable. Approximable applications have one or more tasks that can be replaced by their approximate versions. Such applications are modeled as compound task graphs, as shown in Figure 8. We generate compound task graphs that include multiple versions of approximable tasks, (in dotted lines), the approximate tasks are shown in solid fill.

B. System Architecture

1) Run-time Mapping and Mode Selection: Incoming applications are queued in the application repository and make a request to be executed on the chip. The run-time mapping and mode selection unit (RMSU) responds to the application request by selecting free cores and maps the application in a task-per-core manner. In addition, if the application is approximable, RMSU buffers the approximate tasks from the compound task graph into the *Task Bank*. The *Task Bank* is implemented as a memory structure that holds pointers to the addresses of approximable tasks, to provide easier access to approximable tasks without a significant overhead. We use pro-active application mapping *MapPro*, presented in [27]. Upon application mapping, RMSU sends the core allocation information to the power controller for potential actuation decisions. Servicing an application depends on availability of free cores on the chip, which in turn depends on performance and power consumption of active cores. Number of outstanding application requests weighed with the time before they get serviced, AWT (Equation 1), is sent to the power controller for knob actuation decisions.

2) Power Controller: Power controller is the central manager that monitors power consumption, incoming application request rate and system metrics for power and performance knob actuation decisions. Every core on the chip is provided with power and processor utilization sensors. We use the combination of processor utilization, packet injection rate and buffer utilization to prune the design space for selection of candidates that are more suitable for voltage down/up scaling. Selection of appropriate candidates for employing the DVFS and PG knobs are elaborated in our previous work [5] [17]. Thus we monitor power consumption, processor utilization, network congestion and network intensity at run-time, forming the *monitor* phase of the power management framework. Actuation decisions of the power controller are based on parameters received from the *monitor* phase. We feed the difference between power consumption of the chip and TDP to a PID controller. Output of the PID controller is proportional to the difference between power consumption and TDP and determines voltage and frequency levels to be down scaled to avoid power violation. In case of power consumption being below TDP, voltage and frequency would be up-scaled for better power utilization. *Knob Setting* block of the power controller receives the new voltage and frequency levels from the PID controller, along with processor utilization, buffer utilization and packet injection rate from the *monitor* phase. Based on utilization and network parameters, *Knob Setting* block decides the cores to which voltage and frequency levels are to be updated. The PID controller's output is also used to decide the number of cores to be power gated. Both DVFS and PG actuations are applied to the chip, as shown in Figure 7.

Load analyzer compares application request rate, represented by AWT, and the threshold, AWT_{th}, to determine throughput violation ($AWT > AWT_{th}$). The *Knob Setting* uses this information and invokes APPX knob. Sensitivity metric over different levels of approximation for all the applications is summarized into a look-up table. Each application has k (parametrized) levels of approximation and sensitivities associated with each level. The level of approximation of an application that is currently running on the chip determines current sensitivity factor, while the ones that are preceding and succeeding are the previous and next sensitivity factors. The look-up table is pruned to find the application that has the

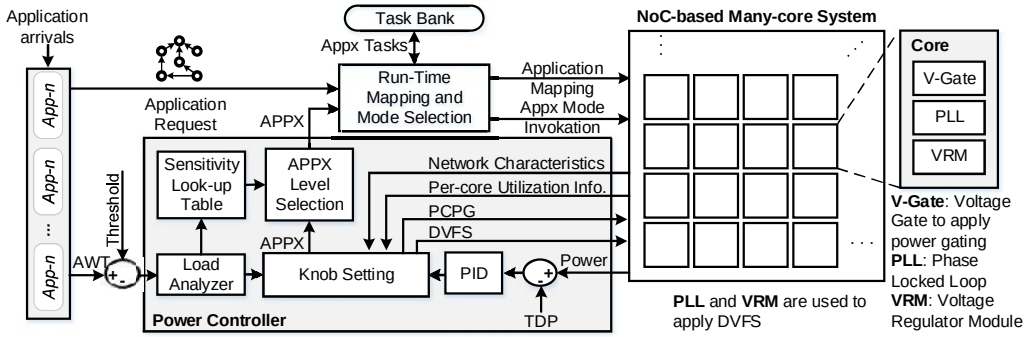


Figure 7: Power Management Framework

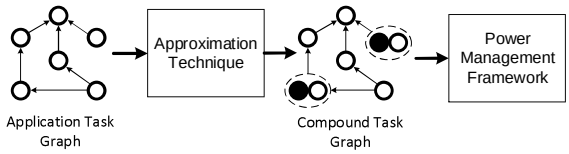


Figure 8: Compound Task Graph - Workflow

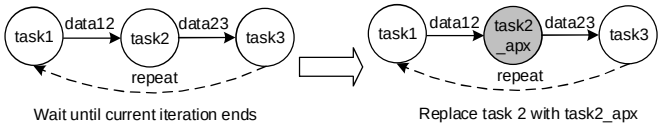


Figure 9: Mode Switching

highest sensitivity factor in its next level of approximation. For example, consider App_1 running at level-1 of approximation and app_2 running at level-2 of approximation. If APPX is to be invoked, the next level of approximation for app_1 is level-2 and for app_2 is level-3. So, the sensitivities of app_1 at level-2 (succeeding the current level-1) and app_2 at level-3 (succeeding the current level-2) are compared to find the application with highest *next sensitivity* value. The chosen application and corresponding level of approximation are forwarded to the RMSU. The RMSU retrieves the approximable tasks of the chosen application with the level specified by the *APPX Level Selection* from the Task Bank. The accurate task is then replaced with the approximate task retrieved from the *Task Bank*. For evaluation, we currently use four levels of approximation in increasing order of accuracy-performance trade-offs. Alternatively, several fine-grained levels of accuracy trade-offs could be used.

Mode Switching: APPX knob invocation triggers switching the mode of execution of an approximable application. Depending on APPX knob setting, 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. The level of approximation is specified by the power manager. For applications that are currently running on the chip, power manager chooses the application(s) and level of approximation to switch to. Based on these, RMSU identifies the corresponding approximate task specified by the power manager from the *Task Bank* 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. In a similar manner, IoT applications work on real world sensory data, pre-processing, filtering and computation over continuous intervals in a batch. Precisely, in these applications, the computational task remains the same while new data arrives after every interval. When the 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 9 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 Bank*. 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 (I-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 instruction cache used, instructions of task2 may require flushing, however this is subject to 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 (D-cache) is not needed. The state of the application is hence preserved at the end of the period. The mode switching overhead is elaborated in Section VII.

C. Power Management Algorithm

We employ the DVFS and PG knobs synergistically with APPX in a hierarchical way, as shown in Figure 10. Triggering and tuning of these knobs together for power capping and performance maintenance is handled by power management

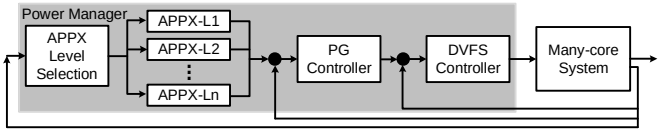


Figure 10: Hierarchy of Knobs

Algorithm 1 Power Management Algorithm

Inputs: P: Power Consumption, AWT: Accumulated Wait Time;
Outputs: $APPX$, $PCPG$, $DVFS$, $mode$: Knob actuations and Mode switching commands for RMSU;
Constants: TDP: Power budget, AWT_{th}: *AWT* Threshold, $e1$, $e2$, $e3$: Knob actuation epochs, $SFLT$: Sensitivity factor look-up table
Global Variables: $currSF[\,]$, $nextSF[\,]$, $prevSF[\,]$: Sensitivity factor vectors of current, next and previous levels of approximation of running applications; $cores_{un}$, $cores_{gate}$, $cores_{pref}$: Cores - Un-occupied, power gated and preferable for DVFS.
Variables: App_R, App_X, App_Y: Applications - currently running, approximable, running in approximate mode; SF: Sensitivity factor;

Body:

```
1: for epoch = e1 do
2:    ΔT = AWT − AWT_th;
3:    (app, level, mode) = appxChoose(ΔT, currSF);
4:    APPX(app, level, mode);
5:    for epoch = e2 do
6:       ΔP = P − TDP;
7:       if ΔP ≥ 0 then
8:          PCPG_gate(cores_un);
9:       else
10:         PCPG_un(cores_gate);
11:      for epoch = e3 do
12:         ΔP = TDP − P;
13:         if ΔP ≥ 0 then
14:            DVFS_down(cores_pref);
15:         else
16:            DVFS_up(cores_pref);
```

Algorithm 2 Approximation level calculation function ($appxChoose()$)

Inputs: ΔT, $currSF(SF[\,])$;
Outputs: app: Application chosen for approximation, $level$: level of approximation, $mode$: Mode for incoming applications;

Body:

```
1: if ΔT ≥ 0 then
2:    if ΔT ≥ 1 then
3:       mode = in;
4:    else
5:       mode = run;
6:    SV[ ] ← nextSF(SFLT[ ]);
7:    sort(SV[ ]);
8:    (app, level) = max(SV[ ]);
9: else
10:   SV[ ] ← prevSF(SFLT[ ]);
11:   sort(SV[ ]);
12:   (app, level) = min(SV[ ]);
13: return (app, level, mode);
```

Algorithm 3 Mode Switching Algorithm

Inputs: app: Application chosen for mode switching, $level$: Level of approximation, $mode$: Mode of mapping a new application
Outputs: Map: Mapping configuration of incoming application;
Variable: $newApp$: Incoming application, t^{level}: Approximable task and its level of approximation;

Body:

```
1: if mode = in then
2:    if newApp then
3:       for t^a ∈ newApp do
4:          TaskBank.push(t^a);
5:          switch(t^1, t);
6:          Map(newApp);
7: if mode = run then
8:    wait until current iteration of t finishes;
9:    switch(TaskBank→ app.t^level, app.t);
```

algorithm, listed in Algorithm 1. We define three epochs e_1, e_2 and e_3 for actuation of APPX, PG and DVFS knobs respectively, such that $e_1 > e_2 > e_3$. At every epoch e_1, application request rate is monitored. The difference between AWT and its threshold is used to choose settings for APPX knob. Settings for the APPX knob viz., application and level of approximation are determined by level selection algorithm, listed in Algorithm 2. The extent of throughput violation (ΔT) determines mode of incoming applications. If ($\Delta T > 1$), it reflects a steeper request rate of applications. Then, the mode is set to *in*, indicating to the RMSU that newly incoming applications are to be mapped in their approximate mode at level-1. If ($1 > \Delta T > 0$), the mode is set to *run* i.e., to switch the execution mode of currently running applications only. In this case, based on sensitivity factors of currently running applications, the sensitivity factor vector for next level of approximation is looked-up from the sensitivity factor look-up table ($SFLT$). This vector is sorted to find the application with highest sensitivity among running applications. The chosen application and level of approximation is returned to the power manager for invocation of APPX knob. In case the throughput is not violated ($\Delta T < 0$), the approximation level of a chosen application is shifted up, to a relatively accurate level. The sensitivity factors of preceding level of approximation of currently running applications are looked-up from $SFLT$. These are sorted to find the application that is least sensitive to error. This application is chosen for invocation of APPX knob, the application and its level of approximation are returned to the power manager. APPX knob is invoked by the power manager by sending the knob settings received from the level selection algorithm to the RMSU. This is presented in a listing in Algorithm 3. When the *mode* is *in*, new incoming application's tasks are buffered into the Task Bank. The new application is mapped in its approximate mode at level-1. When the *mode* is set to *run*, the task and its level of approximation specified by the power manager is retrieved from the Task Bank. Accurate version of this task is replaced by the approximate task. Epochs e_2 and e_3 are smaller than e_1 and are concerning power violations. Power violations are monitored at epoch e_2. If power consumption exceeds TDP, PCPG knob is actuated by power gating cores that are currently un-occupied on the chip. Conversely, when power consumption is below TDP, cores that are previously power-gated are powered up. At epoch e_3, power violations are addressed by DVFS knob. Cores that benefit relatively higher

from DVFS actuation are the preferable cores. DVFS knob is actuated over these preferable cores. Similar to the PCPG knob, voltage and frequency levels of preferable cores are up-scaled when power consumption is below TDP. The actuation of PCPG and DVFS knobs are based on our prior work on multi-objective power management framework [5], [17].

VII. Evaluation

In this section, we assess the efficiency of our approximation-enabled power management approaches APPEND+ and APPEND, with and without considering sensitivity of application. We compare our approach against state-of-the-art dynamic power management/capping techniques PG [12] which is based on PCPG, and MOC [5] which is based on per-core DVFS and PG.

A. Application Setup

For evaluation purpose, we choose inter-disciplinary error resilient application domains of machine learning and signal processing. Further, we selected two applications from IoT domain, given the nature of input data and computations involved. The applications used for evaluation of APPEND+ are presented in Table I. The chosen machine learning applications are data-triggered on-line learning techniques that fall under classification and estimation. They are inter-disciplinary, specifically with IoT based applications, being used in recognition, mining, synthesis 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. For evaluation purpose, we choose 4 levels of approximation, level-1 through level-4. We normalize the performance and energy gains of approximate tasks from level-1 to level-4 against their accurate versions. Table I shows the normalized gain in performance and energy, relative error induced with approximation for different applications and levels of approximation. The applications tested are error resilient in general. We chose approximations that result in soft errors, ensuring there are no critical errors or exceptions. For linear regression, we use training data of 1 million data samples and a test it over 1000 samples. We use loop perforation to skip 5% to 15% of computations on input training data. The error is calculated as relative to accurate regression. For k-means clustering and k-nearest neighbors, we use relaxed convergence. 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 from 3% to 10% for four levels of approximation. For FFT, we approximate the computation involving exponential functions with relaxed memoization and storing the twiddle factors in lower precision. We compute the complex exponential for one iteration and re-use it for subsequent samples, despite the inputs to exponential function not being same. For low-pass filter, we use a Blackman window with 50 coefficients. We reduce the number of coefficients up to 5 for relaxed execution. We use sparse vector multiplication because of its broader usage and application in several other fields. For this application, we simplify the logic of product calculation that replaces accumulated multiplications of rows and columns with a single multiplication of means of a row and column. For IoT case study on EWS, we reduce the number of samples of heart rate sensor and compromise data fusion. We run the accurate and approximate versions on data sets collected from 3 different subjects. For fall detection, we reduce the sampling rate of accelerometer, number of filter coefficients and simplify the logic of magnitude vector computation. We use real time accelerator data collected from a subject wearing the fall detector, over different physical activities of walking, sprinting and resting.

We used the normalized performance gain and relative error induced upon mode switching for each applications over 4 levels of approximation. We calculated the error sensitivity as gain in performance per error, as described in Equation 2. Table II shows the sensitivity metrics for the applications used over different levels of approximation. It should be noted that normalized gain and sensitivity with increasing level of approximation are distinct. The sensitivity metric represents amount of performance gained per amount accuracy lost by moving a level of approximation further. For example, linear regression has sensitivity of 10.1 at level-1, while the sensitivity at level-2, 0.09, is much lower than the previous level. This intuitively means that changing the level of approximation for this application from level-1 to level-1 either has a lower performance gain or higher error penalty or both. Similarly, k-means at level-1 has much higher sensitivity than that of the other levels. The normalized gain at level-1 for k-means is 1.23, however the error, 0.01, (see Table 1) is extremely small, making the sensitivity high. APPEND+ considers the sensitivity metric of all the applications currently running on the chip to make decisions on which application and which level of approximation are to be chosen for APPX knob actuation. APPEND is oblivious to the sensitivity metric and chooses applications in a naive manner and corresponding level of approximation in a sequentially increasing order.

B. Simulation Environment

Applications are modeled as task graphs, as described in Section VI. We implement each application such that one task is allocated one core on interval-core based Sniper simulator, annexed with McPAT for modeling power [28]. We used Gainestown architecture that has Nehalem-like processing elements with 32KB of instruction and data caches. We model the application into 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 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. The performance and power values extracted from these simulations provide the relative performance and power gains of a task when the execution is switched to approximate from accurate. We use this ratio to model the power and throughput gains while simulating, so that the APPEND+ framework can be adaptive for all hardware platforms irrespective of architecture.

Table I: Applications' Energy-Accuracy Trade-offs

Application		Norm. Perf.				Norm. Energy				%Error			
		Level-1	Level-2	Level-3	Level-4	Level-1	Level-2	Level-3	Level-4	Level-1	Level-2	Level-3	Level-4
Machine Learning	Linear Regression	1.01	1.09	1.15	1.20	1	1.08	1.15	1.2	0.01	0.05	0.08	0.1
	K-means	1.23	1.54	1.71	1.92	1.22	1.53	1.69	1.89	3.53	12.74	16.34	19.66
	K-NN	1.08	1.1	1.15	1.57	1.11	1.13	1.18	1.75	1.13	2.1	5.85	28.8
Signal Processing	FFT	1.01	1.01	1.05	1.07	1.04	1.06	1.08	1.11	2.17	4.14	7.6	13.34
	LPF	1.1	2.0	3.34	10.5	1.2	2.0	3.5	9.33	15.72	16.6	19.35	30.09
	Vector Multiplication	1.07	1.18	1.25	1.68	1.06	1.15	1.2	1.6	8.53	12.39	18.2	30.0
IoT	EWS	1.11	1.15	1.33	1.49	1.22	1.29	1.37	1.57	10.4	14.5	17.49	27.72
	Fall Detection	1.1	1.39	1.7	2.06	1.1	1.4	1.75	2.33	9.18	11.38	14.1	22.9

The Sniper simulator provides processing elements with other variants of micro-architecture. The relative performance gains for approximate versions over the accurate versions may hold good over different hardware platforms, unless the architecture is highly customized. We use our in-house cycle accurate simulator implemented in SystemC to evaluate the proposed power management framework. We extended Noxim [29] NoC simulator using its network infrastructure for interconnects. The power characteristics of processing elements (PE) are modeled based on metrics extracted from McPAT and Lumos [30]. 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 [27] 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 [12] and [5]). The TDP value is set to 90W, calculated based on the chip's power density. We also evaluate our approach for power capping under a variable power budget, thermal safe power (TSP) [4]. TSP is calculated as a function of simultaneously active cores, which vary at run-time based on application arrival and mapping. TSP provides a relatively higher power budget than the conservative design time estimate of TDP. We estimate TSP on-line and use it as the upper bound on power budget for evaluating APPEND+. We implemented the APPEND+ technique over integrated simulation framework as summarized above. It is possible to implement the same as an operating system level policy, provided the hardware platform supports sensing and actuation of power.

C. 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 goal is to cap the power consumption such that TDP constraint is honored throughout the period of execution. Figure 13 shows the power consumption of DVFS, PG, MOC, APPEND 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, APPEND and APPEND+. Figure 14 shows the power consumption of the system with TSP as the upper bound on power. Unlike TDP, TSP varies at run-time offering a flexibility in power capping. DVFS knob violates the TSP limit and is not at efficient power capping. PG honors TSP constraint, however the power consumption always remains lower than (but not closer to) TSP, indicating a lower utilization of available power budget. MOC, APPEND and APPEND+ meet the power capping requirement and have a better power budget utilization. APPEND and APPEND+ maintain 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 and APPEND+ are able to service applications faster, reducing the run-time and consequently wait-time of incoming applications. Figure 15 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 APPEND, MOC, PG and DVFS. DVFS and PG based actuations have higher AWTs already when

Table II: Application Sensitivity

Application		Sensitivity			
		Level-1	Level-2	Level-3	Level-4
Machine Learning	Linear Regression	10.1	0.09	0.13	0.23
	K-means	6.51	0.28	0.44	0.64
	K-NN	0.07	0.06	0.06	0.43
Signal Processing	FFT	0.01	0.01	0.05	0.07
	LPF	0.6	5.5	2.9	1.84
	Vector Multiplication	0.24	1.59	0.31	0.83
IoT	EWS	0.95	0.26	0.58	0.3
	Fall Detection	1.02	0.86	0.72	0.14

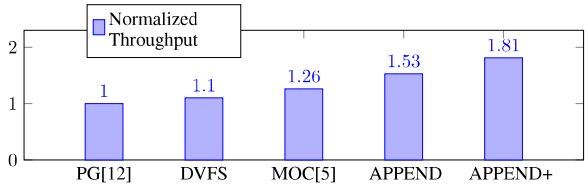


Figure 11: Normalized throughput (TDP)

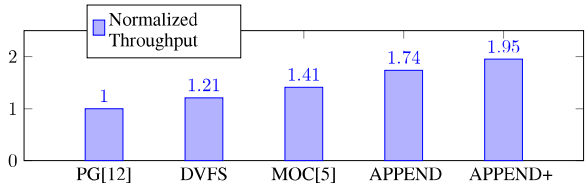


Figure 12: Normalized throughput (TSP)

the application request rate reaches 3 per second. MOC has a relatively high AWT when application request rate is 5 per second. However, APPEND and APPEND+ have 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 and APPEND+ 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 15 per second. Also, *AWT* accumulation is more steeper in case of other knobs than that of APPEND and APPEND+, indicating a substantial rise in their wait-times with high request rates. The minimal *AWT* and high service rates of APPEND and APPEND+ also results in high throughput and energy efficiency. Normalized gain in throughput for all knob combinations with TDP as upper bound and TSP as upper bound are shown in Figure 11 and Figure 12 respectively. APPEND+, followed by APPEND have the higher throughput, that is upto 1.9× better than PG and 1.4× 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. It is also to be noted that throughput gain of APPEND+ is relatively higher with TSP than that of TDP. The same trend can be seen with throughput of all the other knobs too, reflecting better utilization of power under TSP constraint. APPX knob can be implemented exclusively in software, making the APPEND+ framework scalable and adaptable across different hardware platforms. In the context of IoT applications, APPEND+ can be used both at the edge and cloud layers which can deliver real-time high performance at the front-end. The sensitivity look-up table can be used only to store sensitivity metrics of applications that are currently running on the chip, limiting the size of the look-up table without affecting scalability.

1) Error and Overhead Analysis: Behavioral patterns of accurate (Acc) and approximate (Appx) versions of each application are shown in Table III. The approximate versions are at level-4 of approximation, to reflect the maximum overhead caused and maximum performance gained. The number of instructions of each applications (Instructions), number of

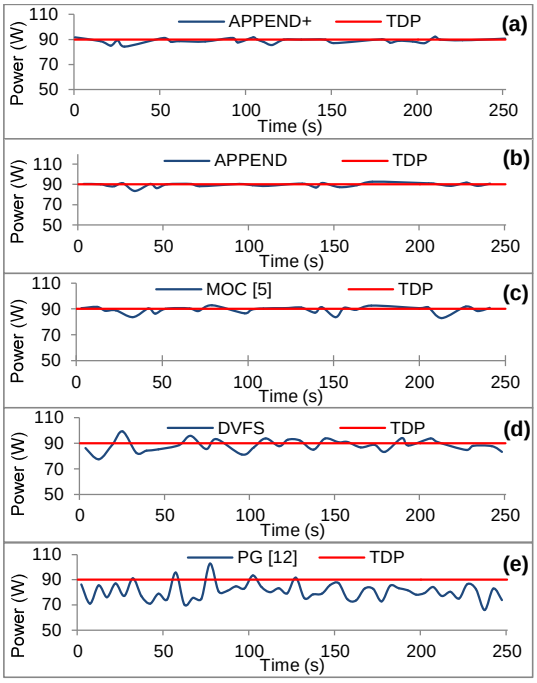


Figure 13: Instantaneous Power Consumption of different knobs on 144-core system. Power is capped at TDP.

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 III. These metrics are extracted from individual application simulations (accurate and approximate) on Sniper. 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. 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 in mode switching 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. In terms of power overhead, the *Task Bank* and sensitivity look up tables used in APPEND+ framework are simple memory structures with fewer access during execution of an application. The power consumption of these components is insignificant compared to the total system power. 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

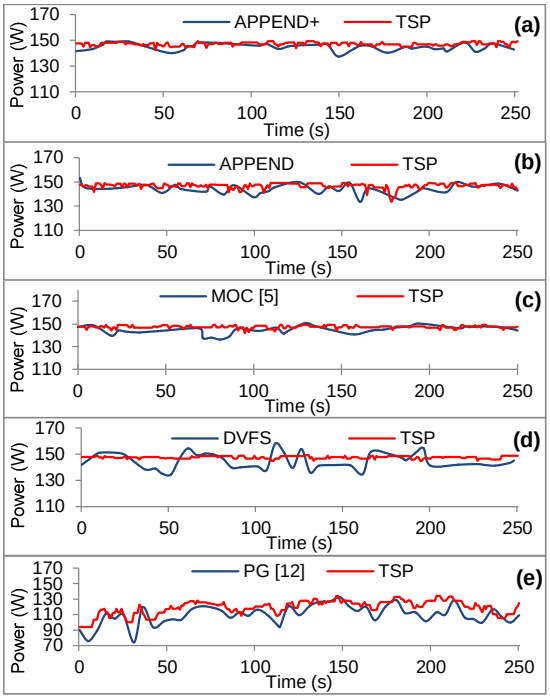


Figure 14: Instantaneous Power Consumption of different knobs on a 144-core system. Power is capped at TSP calculated at run-time

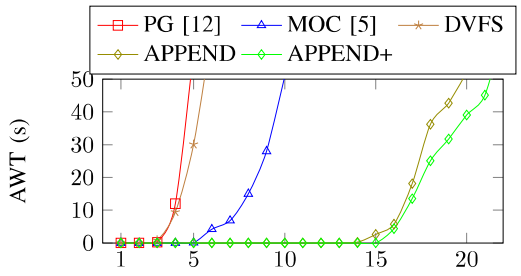


Figure 15: Accumulated waiting time

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. 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:

$$penalty = \frac{size_{app}}{size_{pkt}} \times (P_L + (n \times r_L) + MC_L + DRAM_L) \quad (3)$$

where $size_{app}$ and $size_{pkt}$ are application and packet sizes, P_L is packetizing latency, 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 the theoretical worst case overhead penalty of mode switching for a video encoding application run on Intel SCC as an example [31]. The

Table III: 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
FFT	1147	1159	251.4	249	37	36	0.8
LPF	721	721	197	18	19.9	19	0.1
Vector Multi-plication	775	779	33.2	22.7	20.3	19.8	0.5
EWS	1017	1032	66.1	4.1	4.3	4.1	0.3
Fall Detection	795	801	41.9	20.05	3.4	2.23	0.1

experimental many-core platform Intel SCC has the off-chip memory, core and network frequencies of 400MHz, 533MHz and 800MHz respectively. The worst case mode switching penalty on SCC for the video encoding application of size 6KB using the formula in Equation 3 is 1.5ms. For the same application, penalty in task migration is 10.6ms, 7× more than the mode switching overhead, 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.

VIII. Conclusions

In this work, we proposed approximation as another knob for power management in many-core systems. We implemented APPX knob based on application's sensitivity to error such that performance gain is maximized within minimal error. We developed power managing schemes to combine DVFS and PG knobs with APPX knob to meet system requirements in power capping, performance and energy efficiency. We presented a power management framework, APPEND+, that monitors chip's power and performance requirements at run-time and triggers different knob actuations accordingly. We evaluated APPEND+ against other state-of-the-art power management techniques, over machine learning, signal processing and IoT applications. APPEND+ improves performance and energy efficiency with the APPX knob and sensitivity aware actuation of the APPX knob, while power capping is maintained with combination of APPX, DVFS and PG knobs.

Acknowledgement

The authors wish to acknowledge the financial support by the Marie Curie Actions of the European Union's H2020 Programme.

References

[1] A. Wolgang *et al.*, "More than Moore," in *ITRS, 2010*, 2010.

[2] A. Rahmani *et al.*, *The Dark Side of Silicon*, 1st ed. Springer, Switzerland, 2016.

[3] H. Esmaeilzadeh *et al.*, "Dark silicon and the end of multicore scaling," in *ISCA*, 2011.

[4] S. Pagani *et al.*, "TSP: Thermal Safe Power: Efficient Power Budgeting for many-core systems in dark silicon era," in *CODES+ISSS*, 2014.

[5] A. Rahmani *et al.*, "Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach," in *ISLPED*, 2015.

[6] K. Ma and X. Wang, "PGCapping: Exploiting power gating for power capping and core lifetime balancing in CMPs," in *PACT*, 2012.

[7] L. Wang and K. Skadron, "Implications of the power wall: Dim cores and reconfigurable logic," in *IEEE Micro*, 2013.

[8] N. Kapadia *et al.*, "VARSHA: Variation and Reliability-aware Application Scheduling with Adaptive Parallelism in the Dark-silicon Era," in *DATE*, 2015.

[9] T. Komoda *et al.*, "Power capping of CPU-GPU heterogeneous systems through coordinating DVFS and task mapping," in *ICCD*, 2013.

[10] A. M. Rahmani *et al.*, "Smart e-health gateway:bringing intelligence to internet-of-things based ubiquitous healthcare systems," in *CCNC*, 2015.

[11] A. Kanduri *et al.*, "Approximation knob: Power capping meets energy efficiency," in *ICCAD*, 2016.

[12] M.-H. Haghbayan *et al.*, "Dark Silicon Aware Power Management for Manycore Systems under Dynamic Workloads," in *ICCD*, 2014.

[13] A. Vega *et al.*, "Crank it up or dial it down: Coordinated multiprocessor frequency and folding control," in *MICRO*, 2013.

[14] R. Cochran *et al.*, "Pack & cap: adaptive dvfs and thread packing under power caps," in *MICRO*, 2011.

[15] A. Kanduri *et al.*, "Dark silicon aware runtime mapping for many-core systems: A patterning approach," in *ICCD*, 2015.

[16] M. Shafique *et al.*, "Dark Silicon As a Challenge for Hardware/Software Co-design," in *CODES+ISSS*, 2014.

[17] A. Rahmani *et al.*, "Reliability-aware runtime power management for many-core systems in dark silicon era," in *IEEE Transactions on Very Large Scale Integration Systems*, 2016.

[18] H. Chen *et al.*, "Dynamic server power capping for enabling data center participation in power markets," in *ICCAD*, 2013.

[19] J. Ansel *et al.*, "PetaBricks: a language and compiler for algorithmic choice," *ACM SIGPLAN Notices*, 2009.

[20] W. Baek *et al.*, "Green : A Framework for Supporting Energy-Conscious Programming using Controlled Approximation," in *PLDI*, 2010.

[21] H. Hoffmann *et al.*, "Dynamic knobs for responsive power-aware computing," *ACM SIGPLAN Notices*, 2012.

[22] S. Sidiroglou *et al.*, "Managing performance vs. accuracy trade-offs with loop perforation," in *FSE*, 2011.

[23] T. N. Gia *et al.*, "Fog computing in healthcare internet of things: A case study on ecg feature extraction," in *Proc. of CIT*, 2015.

[24] C. Tan *et al.*, "Locus: low-power customizable many-core architecture for wearables," in *Proc. of CASES*, 2016.

[25] A. Anzanpour *et al.*, "Internet of Things Enabled In-Home Health Monitoring System Using Early Warning Score," in *MobiHealth*, 2015.

[26] A. Odunmbaku *et al.*, "Elderly Monitoring System with Sleep and Fall Detector," in *HealthyIoT*, 2015.

[27] M. Haghbayan *et al.*, "MapPro: Proactive Runtime Mapping for Dynamic Workloads by Quantifying Ripple Effect of Applications on NoCs," in *NOCS*, 2015.

[28] E. Trevor *et al.*, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," in *SC*, 2011.

[29] Fazzino *et al.*, "Noxim: Network-on-chip simulator," *URL: http://sourceforge.net/projects/noxim*, 2008.

[30] L. Wang and K. Skadron, "Dark vs. dim silicon and near-threshold computing extended results," *Univ. of Virginia, Dept of Comp.Sci Technical Report*, vol. 1, 2012.

[31] S. Holmbacka *et al.*, "A task migration mechanism for distributed many-core operating systems," *Journal of Supercomputing*, vol. 68(3).

Anil Kanduri received B.Tech degree in Electronics and Communications from JNTU Kakinada, India and M.Sc (Tech) in Embedded Computing from University of Turku, Finland. He is a PhD student at the department of Information Technology in University of Turku, since 2014. His interests are in energy efficient computer architectures, run-time management and approximate computing.

Mohammad-Hashem Haghbayan received the BA degree in computer engineering from Ferdowsi University of Mashhad and the MS degree in computer architecture from University of Tehran, Iran. Since 2014 he is PhD student in University of Turku, Finland. His research interests include high-performance energy-efficient architectures, power management techniques, and online/offline testing. He has several years of experience working in industry as well as developing research tools before starting his PhD.

Amir M. Rahmani received his MSc degree from University of Tehran, Iran, in 2009 and Ph.D. degree from University of Turku, Finland, in 2012. He holds an MBA from Turku School of Economics and European Institute of Innovation & Technology (EIT) ICT Labs, in 2014. He is currently Marie Curie Global Fellow at University of California Irvine (USA) and TU Wien (Austria), and also adjunct professor (Docent) in embedded parallel and distributed computing at the University of Turku, Finland. He is the author of more than 120 peer-reviewed publications.

Pasi Liljeberg received MSc and PhD degrees in electronics and information technology from University of Turku, Finland, in 1999 and 2005, respectively. He is a full professor in embedded computing architectures at the University of Turku, Embedded Computer Systems laboratory. His current research interests include parallel and distributed systems, Internet-of-Things, e-Health, embedded computing architecture, fog computing, 3D multiprocessor system architectures, cyber-physical systems, and reconfigurable system design.

Axel Jantsch received the Dipl.-Ing. and Dr.Tech. degrees from the Technische Universität Wien, Vienna, Austria, in 1988 and 1992, respectively. From 1997 to 2002, he was an Associate Professor with the KTH Royal Institute of Technology, Stockholm, Sweden, where he was also a Full Professor of Electronic Systems Design from 2002 to 2014. Since 2014, he has been a Professor with the Institute of Computer Technology, TU Wien. He has authored over 300 articles and one book in the areas of VLSI design and synthesis, HW/SW codesign and cosynthesis, networks-on-chip, and self-awareness in cyber-physical systems.

Hannu Tenhunen received the diplomas from the Helsinki University of Technology, Finland, 1982, and the PhD degree from Cornell University, NY, 1986. In 1985, he joined the Signal Processing Laboratory, Tampere University of Technology, Finland, as an associate professor and later served as a professor and department director. Since 1992, ha has been a professor at the KTH Royal Institute of Technology, Sweden, where he also served as a dean. His current research interests are VLSI architectures and systems, especially network-on-chip systems.

Nikil Dutt received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Urbana, in 1989. He is currently a Chancellor's Professor of computer science with the Department of Electrical Engineering and Computer Science, University of California, Irvine. His current research interests include embedded systems, electronic design automation, computer architecture, systems software, formal methods, and brain-inspired computing. Dr. Dutt is an ACM Distinguished Scientist and an IFIP Silver Core Awardee.