Workload- and Process-Variation Aware Voltage/Frequency Tuning for Energy Efficient Performance Sustainability of NTC Manycores

Ioannis Stamelakos[a], Sotirios Xydis[b], Gianluca Palermo[a], Cristina Silvano[a]

[a]*Politecnico di Milano - Dipartimento di Elettronica, Informazione e Bioingegneria*
[b]*Institute of Communication and Computer Systems - National Technical University of Athens*

Abstract

The power-wall raised by the stagnation of supply voltage in deep-submicron technology nodes, is now the major scaling barrier for moving towards the many-core era. At the same time, the adoption of manycore architectures is considered to be crucial for satisfying the increasing computational power demands and throughput requirements imposed by the explosion in software complexity and volume. The rise of the so-called Dark Silicon, caused by the power budget violations that allow only a small portion of the available computational resources to be simultaneously exploited, points to the direction of energy efficient platforms. Near-Threshold voltage Computing (NTC) has emerged as a promising approach to overcome the manycore power-wall, at the expense of higher sensitivity to process variation and reduced performance which can be compensated with massive parallelization. Given that several application domains operate over specific performance constraints, the performance sustainability is considered a major issue for the wide adoption of NTC. In this work, assuming a feasible, low overhead Power Delivery Network (PDN) for NTC, we investigate how performance guarantees can be ensured when moving towards NTC manycores through a variability-aware voltage and frequency allocation methodology, showing that performance can be efficiently sustained at the NT region while reducing energy dramatically. Additionally, we propose an algorithm for balancing throughput un-

Email addresses: ioannis.stamelakos@polimi.it (Ioannis Stamelakos), sxydis@microlab.ntua.gr (Sotirios Xydis), gianluca.palermo@polimi.it (Gianluca Palermo), cristina.silvano@polimi.it (Cristina Silvano)

der process (and workload) variation that sustains performance while providing significant energy savings.

Keywords: Near-Threshold Computing, Manycore Architectures, Low Power, Energy Efficiency, Variability

1. Introduction

Technology scaling has enabled the chip industry to integrate more cores per processor leading to manycore designs. However, the stagnation of voltage scaling, known also as Dennard's law, and its thermal and power budget implications, do not let us exploit the full potential of those architectures. This problem, widely known as Dark Silicon, is expected to be exacerbated in the future. At the same time the demand for computational power is steadily increasing even though it is hidden from the average user through cloud computing and server applications. Manycore processors can provide high throughput for highly parallel workloads, making them ideal candidates for running multi-threaded parallel applications, however, they are constrained by their strict power/thermal budgets.

Near-Threshold Voltage Computing (NTC) [1] represents a promising technique to mitigate the effects of dark silicon in manycore architectures, allowing a large number of cores to operate simultaneously under a given power envelope. In comparison with the conventional Super-Threshold Voltage Computing (STC), computations at the NT region are performed in a very energy efficient manner, unfortunately at the expense of reduced performance and high susceptibility to parametric process variation. Thus, sustaining performance levels delivered in conventional STC is considered a key factor for enabling a wider adoption of NTC in manycores.

Several studies, e.g. [2], [3], have shown that NTC performance loss can be compensated at some level by squeezing all the available parallelism, however they adopt a best-effort strategy to mitigate process variability without providing any performance guarantees. Recently, voltage island management techniques for NTC performance sustainability have been presented [4], [5]. However, the analysis is performed assuming an ideal power delivery model, i.e. power conversion overheads are not taken into account, and only the effects of process variability are compensated. Moreover, parallel applications are exhibiting also workload variability, originated by unbalanced thread execution. In several studies [6], [7], the authors address workload variability by balancing power consumption while

maximizing throughput. However, the proposed techniques target the ST region, neglecting the increased effects of NTC process variability.

In this work, we address the problem of performance sustainability when an application moves from the ST to the NT region. Unlike previous approaches in NTC [2], [3], [4], [5], we adopt realistic system models accounting for irregularities and overheads imposed by the Power Delivery Network (PDN). Specifically, we propose a voltage and frequency tuning strategy that enables energy efficient and performance sustainable NT manycores through the combined mitigation of workload- and process-dependent variability. The proposed tuning strategy first performs frequency upper bounding, guaranteeing performance sustainability in NTC as well as throughput balancing and then further fine-tunes voltage allocation, mitigating the variability effects. As shown, this technique is extremely lightweight, i.e. easily integrated in the runtime system and provides increased NTC energy efficiency.

Extensive experimental analysis showed than we can have energy gains ranging from 55% up to 76% while sustaining performance and eliminating variability in NTC. Additionally, we can mitigate significantly the throughput imbalance caused by the increased variability in NTC, obtaining in that way an extra 39% of energy savings. The rest of the paper is organized as follows: In Section 2 we present the related research in NTC, in Section 3 we describe extensively the micro-architectural, process variation and power delivery model, in Section 4 we analyze our methodology for sustaining performance and balancing throughput in NTC, Section 5 presents the experimental results and finally 6 concludes this work.

2. Relative Research

2.1. NTC Manycore Architectures

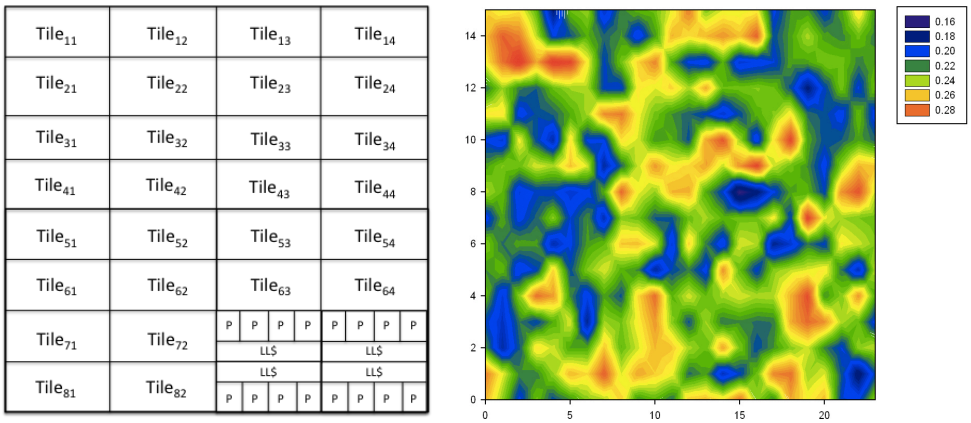
The NTC paradigm has emerged as an extremely optimized solution for energy-efficient systems through low-voltage processing. Early studies [1],[2] have shown that near-threshold voltage operation provides an optimal design point in respect to energy and performance efficiency. Early NT voltage processor prototypes [8], [9] have been recently demonstrated validating the theoretical premises, while several studies show the high efficiency of NTC for cloud- [10] and server-based workloads [11].

Targeting mainly manycore architectures, NTC imposes several challenges regarding the application mapping and the resource/power management due to its increased sensitivity to parametric variation [12]. In [3], Karpuzcu et al. proposed a

single-voltage multiple-frequency power management scheme for clustered Near-Threshold Voltage (NTV) many-cores, aiming mainly at the minimization of the on-chip power regulation overhead. In [4], the authors are addressing the problem of sustaining performance when moving to NTC. They explore the efficiency of voltage island domains and propose a multiple-voltage single-frequency power management scheme to mitigate within-die variability at NTC voltages. In [5], they extend power management towards a multiple-voltage multiple-frequency, to further improve performance. Recently, the use of machine-learning techniques has been proposed for voltage/frequency allocation in wide-range processors [13], while in [6] the authors proposed a variability aware thread balancing scheme for throughput maximization under power and thermal constraints.

2.2. Power Delivery in NTC

Recently, many researchers have investigated the possibility of delivering near-threshold voltages on-chip: Hsieh et. al [14] designed a linear voltage regulator (VR) with a variable output voltage that ranges from 0.5 to 1 V in steps of 0.1 V. Lee et al. [15] evaluated the technique known as "voltage stacking" for near-threshold voltages: multiple low-voltage blocks are powered from a single higher voltage by "stacking" the logic blocks and recycling charge between the layers. SuperRange [16] is a wide operational range (0.4 - 1.2 V) power delivery scheme that uses an off-chip VR to deliver the Super Threshold (ST) voltages and an on-chip VR for delivering the NT ones with an average of 70% power efficiency. Booster [17] uses two power supply rails at 400mV and 600mV respectively and uses hints provided by synchronization libraries to determine which cores should be "boosted" (run at higher frequency) maintaining an average per-core frequency and reducing the effect of process variation. In [18], the authors show how the workload characteristics of different applications can be analyzed at design-time and exploited at runtime in order to obtain optimal voltage allocation in NTC, while using standard, low-cost power delivery architectures. Interesting power delivery (PD) architectures with per-core DVFS support have been proposed as well even though they target multicore platforms that do not operate in the NT region. VRCon [19] uses a switching network that allows the cores to share on-chip voltage regulators according to the DVFS intervals and load conditions, improving the overall energy efficiency, but it is not highly scalable since the integration of hundreds of switching (SW) regulators and switches on-chip would impose a significant area overhead. Sinkar et al. [20], based on the observation that a Low-Dropout Regulator (LDO) can share its largest component with the Per-Core Power-Gating (PCPG) device and therefore can be integrated with a minimum



(a) ManyCore Architecture (b) V_{th} variation map

Figure 1: Tile-based architecture (a) and V_{th} variation map derived by [22] (b).

area overhead, demonstrated that a multicore platform with per-core voltage domains is implementable, and under certain assumptions, can be as effective as one having per core switching VRs (instead of LDOs). Godycki [14] et al., by deploying on-chip switched-capacitor regulators, design a reconfigurable power distribution network, offering a wide range of voltages, enabling in that way fast, fine-grained, per-core voltage scaling. All of those techniques address part of the problem but they do not offer a clear solution for a per-core, near-threshold voltage power delivery architecture that could be applied to a manycore platform.

3. System Level Model of NTC Manycore

Micro-architectural model: We focus our study on tile-based architectures, including the ones proposed in [1], [3] and [4]. Figure 1(a) shows an abstract view of the tile-based manycore architecture, as well as the different intra-tile organizations. The intra-tile architecture can be customized by varying the number of cores per tile and the memory configuration of the last level cache (LLC). In this paper, each core owns a private instruction and data cache (P$). The LLC (LL$) is shared among 4 cores, composing a tile. The Intel Nehalem [21] processor configuration for the core and the P$ has been adopted.

3.1. Process Variation Model

In order to capture the process variation at NT operation, we integrate the Varius-NTV [22] microarchitectural model within the proposed framework. While

Varius-NTV reuses the spherical distance function in [23] for modeling the intra-die spatial correlations, it heavily extends it by updating the STC micro-architectural delay and SRAM cell models to reflect in a more accurate manner the higher sensitivity of NTC to process variation. Specifically, it (i) calculates gate-delay according to the EKV model [24] , (ii) it incorporates a 8T SRAM cell model for reliable read/write operations in NTC and (iii) it considers a larger set of memory timing and stability failure modes. We used the ArchFP [25] tool to automatically generate the floorplan of the target manycore architectures. Based on the provided manycore floorplan, Varius-NTV generates the corresponding variation maps accounting for the within-die (WID) and die-to-die (D2D) process variations. Figure 1(b) shows a sample instance of its V_{th} variation map.

Assuming B as the set of component blocks in the floorplan and D the set of dies, we define $V_{th}^{(i,j)}, L_{eff}^{(i,j)}$, $i \in B$, $j \in D$ that corresponds to the V_{th} and L_{eff} respectively of the architecture's component i in sample die j. Once extracted, the two parameters are used for allocating to each component the lowest possible $V_{dd}^{(i,j)}$, for sustaining the f_{NTC} frequency constraint, as described in Section 4.1.

3.2. Power Delivery Model

In most of the research conducted around NTC, the power delivery architecture is neglected and it is generally assumed that the Near-threshold voltages can be delivered without any losses. However, the PDN can have a large overhead on the power consumption if not designed and customized for the specific platform.

Generally, the power delivery network can be divided into two components:

1. Off-chip network: one or more power supply rails, powered by off-chip voltage regulators, deliver the appropriate voltages to the chip.
2. On-chip network: a second layer, connected with the off-chip network, consisting of voltage regulators that step down the voltage and deliver it to the cores. The most common types of VRs are:
    - Switching Regulators: They have a very good efficiency (~90%) but they consume a lot of area and it is hard to integrate them on chip.
    - Low Dropout Regulators: An LDO is kind of linear regulator and its efficiency is calculated as follows:

$$\eta_{ldo} = \frac{P_{out}}{P_{in}} = \frac{I_{out}V_{out}}{I_{in}V_{in}} = \frac{I_{out}V_{out}}{(I_{out} + I_q)V_{in}} \tag{1}$$

where I_q is the quiescent current flowing to the ground:

$$I_{in} = I_{out} + I_q \tag{2}$$

In [20], the authors, based on the observation that Per-Core Power-Gating (PCPG) devices augmented with feedback control circuitry can serve as low-cost Low-Dropout Regulators (LDOs), show that power efficiency as high as that delivered by on-chip switching Voltage Regulators (VRs) can be achieved. The per-core area overhead is only ~2%, making the implementation of this design affordable even for a manycore platform. An LDO, as mentioned above is a type of linear regulator, and its efficiency is defined in Equation 1.

The LDOs exhibit current efficiencies up to 99%, making the difference between the input and the output voltage the dominant factor of Equation 1. The lower the difference, the higher the efficiency, however the difference should not drop below a certain threshold because the circuit ceases to regulate its output voltage against any fluctuations in the input voltage. This threshold is defined as the *dropout voltage* and is defined at design time. The main advantages of an LDO are its fast transient response and its low cost, making it a potential candidate for on-chip VR.

Although typical voltage regulation at low V_{dd} levels exhibit large conversion inefficiencies [26], the authors in [27] have shown that designing and implementing an LDO for near- threshold logic circuits can be performed with quite high efficiency, i.e. around 99%, and fast transition times, i.e. hundreds of nanoseconds up to a few microseconds. In this work, we utilise the LDO configuration adopted directly from [27] with the following design characteristics: 0.5V input voltage, 0.34V - 0.45V output voltage range, 0.05V dropout voltage and 98.7% current efficiency. The SRAM voltage cannot be scaled down after a certain point because it leads to major defects and errors and we assume it has its separate constant supply : 500mV in standby mode and 700mV when accessing.

4. Workload and Process Variability Management for Performance Sustainablity in NTC

The effects of process variation are exacerbated in NTC, but except for that, in order to exploit its energy efficiency potential, we should be able to provide performance guarantees to the applications running in an NTC manycore platform, sustaining their ST performance in the ideal case. This becomes more evident if we consider the emerging paradigm of data centers and cloud computing. To further motivate the aforementioned claim, Figure 2 shows the performance distribution for a 128-core NTC manycore that implements the best-effort EnergyS-

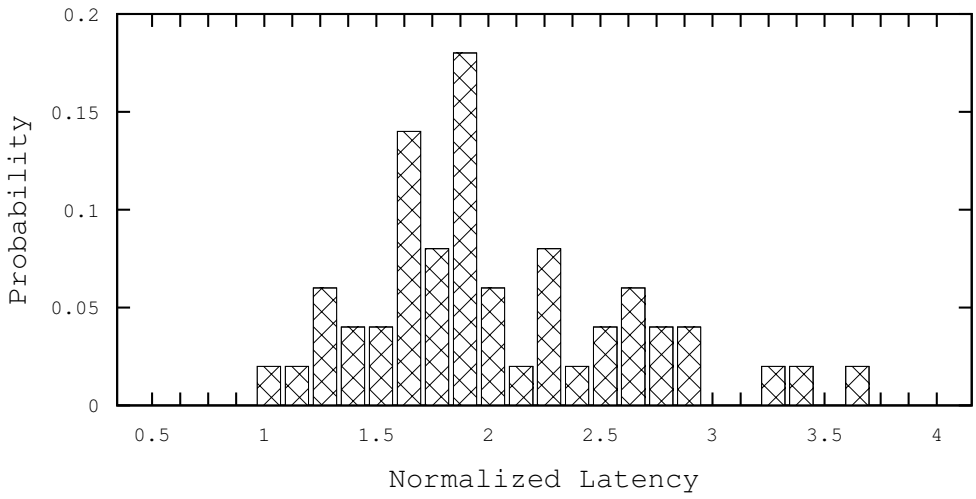


Figure 2: Performance distribution on a 128-core NTC manycore implementing the EnergySmart [3] approach.

mart power management Single-Voltage/Multiple-Frequencies (SVMF) approach [3]. The results are obtained for the execution of the *Barnes* application over 100 different variation maps. The normalized performance value of 1 corresponds to the *nominal* performance of the application. As shown, the performance of NTC manycore platforms is not controllable and is spread out over a wide range of normalized values (from 1 to 3.7) due to the underlying workload and process variability. Thus, the adoption of NTC for applications, exhibiting specific performance and/or throughput constraints, requires careful selection and tuning of the power management scheme. In the following sections, we propose a fast run-time voltage and frequency allocation stategy acounting for both workload- and process- variability, enabling performance sustainability in NTC manycores. The proposed strategy consists of two steps: the first step performs a performance sustainability analysis for generating frequency and voltage upper bounds for NTC operation, while the second step implements the voltage and frequency fine-tuning that enables throughput balancig in NTC, thus mitigating performance variability effects.

4.1. Performance Sustainability in NTC

So far, application workloads have been originally developed and characterized for the ST region. In order to sustain ST performance figures (i.e. latency

or throughput) when moving to the NT region, the inherent parallelism of the applications should be exploited [28] to alleviate the impact of the reduced clock frequencies in NTC. Assuming a minimum allowed latency L_{min} and maximum core count constraint, C_{max} for the NTC manycore, we first calculate the clock frequency of the platform at near-threshold, f_{NTC}, that satisfies the performance constraint. Let $L_{C_{max}}$ be the performance, in terms of latency, of an STC manycore architecture with C_{max} number of cores, running at f_{STC}. When running in STC, $L_{min} - L_{C_{max}} > 0$ is the available latency slack due to the higher degree of parallelism of the architecture, that can be exploited to run the application at lower frequency. Utilizing this positive slack, the f_{NTC} is calculated as follows:

$$f_{NTC} = \frac{L_{C_{max}}}{L_{min}} \times f_{STC} \tag{3}$$

The calculated f_{NTC} refers to the target clock frequency of each core in NTC for sustaining ST performance, without considering the spatial effects of process variations. Next, we extract the values of V_{th} and L_{eff} according to the procedure described in Section 3.1. Finally, the two parameters are used for allocating to each component the lowest possible (provided by the power delivery system) $V_{dd}^{(i,j)}$, for sustaining the f_{NTC} frequency constraint given that:

$$f_{NTC} \propto \frac{(V_{dd}^{(i)} - V_{th}^{(i)})^{\beta}}{V_{dd}^{(i)} \times L_{eff}^{(i)}} \tag{4}$$

where β is a technology-dependent constant (≈ 1.5). The extraction of the f_{NTC} and the per component $V_{dd}^{(i,j)}$, enables the adoption of a fine grained power management scheme for NT operation with guaranteed performance sustainability.

4.2. Throughput Balancing in NTC

After presenting our methodology for sustaining performance in NTC, we move on to dealing with a different but relevant subject: the throughput imbalance observed in the threads of the same application when runnign in NTC. The per-core/thread throughput imbalance that can be caused is usually neglected, however, this cannot be overlooked in NTC, because the underlying variability, which is also exacerbated in each subsequent technology node, has a great impact on both performance and power consumption. In [29], a 30% deviation in frequency among the cores of an Intel 80-core manycore platform has been observed in real, on-silicon measurements. The results are referring to a 65nm technology and a much higher voltage than a near-threshold one, thus the variation is expected to

be much worse at the NT region. In multithreaded parallel applications, there even exists an IPC imbalance caused by the access latency irregularities and the per-thread data workload variation leading to different memory access patterns. Given the experimental setup described in Section 5.1, the per-core, normalized to the minimum value, throughput is shown in Figure 3. The IPCs are obtained from simulations and no process variation is considered, i.e. all the cores run at the same frequency. The standard deviation (σ) is around 11%, but this can be worsened by the actual maximum per-core frequencies caused by process variation. The per-core throughput (normalized to the minimum value), when variability is taken into account, is depicted in Figure 4 for the Cholesky benchmark run on 64 cores. Not only the core exhibiting the least throughput is now a different one but also the variation in per-core throughput has increased dramatically, leading to a standard deviation of 38%.

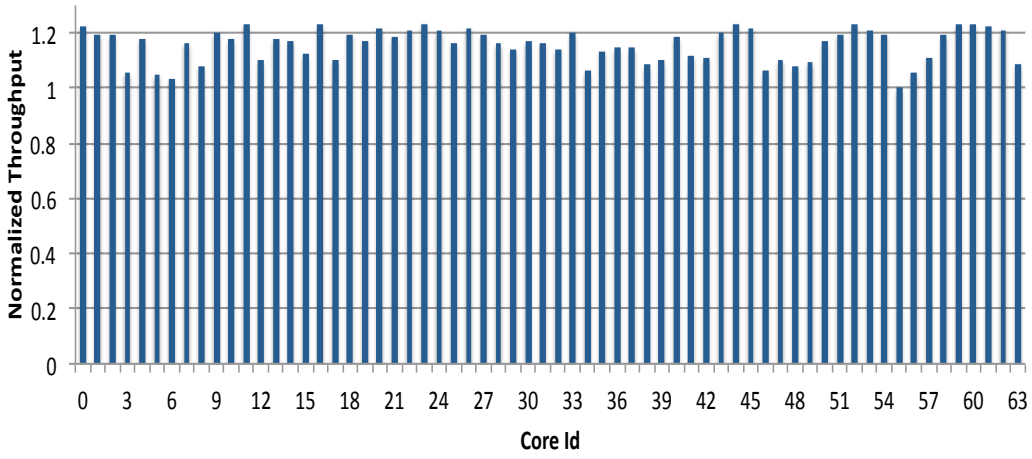


Figure 3: Per-core throughput: imbalance due to IPC variability.

The irregular behavior described above leads us to the conclusion that additional power is wasted from the faster cores even though not needed since they will finish sooner and wait at the synchronization barrier. This observation, creates both a problem that has to be taken care of and an opportunity that should be exploited. Our proposed algorithm manages to mitigate the throughput imbalance and provides up to 43.5% of power savings, on average, for the reported results. Additionally, we are able to identify for each benchmark which is the resource allocation that maximizes energy efficiency, a decision that depends heavily on their unique workload characteristics.

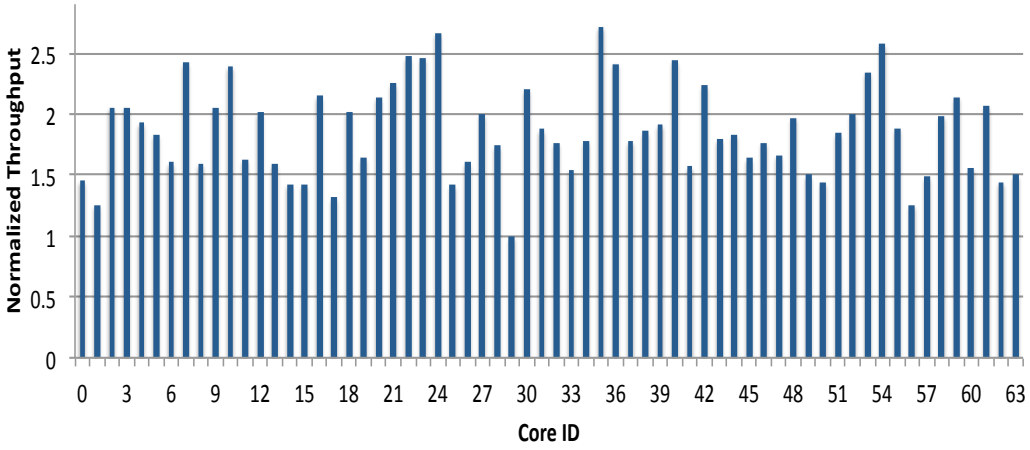


Figure 4: Per-core throughput: imbalance due to process variability.

For this reason we designed an algorithm for dealing with this discrepancy (Algorithm 1). The IPC of each core is provided from our simulations in Sniper and can be obtained at runtime by the performance counters provided by the processor. The relationship between frequency and the transistor characteristics is depicted in Equation 4, where β is a technology-dependent constant (≈ 1.5), V_{th} and L_{eff} is the transistors' threshold voltage and the effective length respectively.

As shown in Figure 4, the throughput achieved among the different cores differs significantly.

Initially, we calculate the maximum per-core throughput that can be achieved assuming that the V_{dd}^i and f_{NTC}, extracted from the previous step, is allocated to them. The throughput of the i_{th} core is calculated as in Equation 5.

$$Throughput_i = IPC_i \cdot F_i \tag{5}$$

where IPC_i and F_i are the Instructions Per Cycle (IPC) and Frequency of the i_{th} core respectively.

In the pseudocode, this first phase is shown in lines 1-4: given the LDO voltage level that satisfies the V_{dd}^i and the f_{NTC} are assigned to the core (line 2-3) and the equivalent throughput is calculated (line 4) for each core. The core exhibiting the minimum throughput is now found and becomes the reference value and bottleneck (line 6). For the rest of the cores, the minimum frequency (and hence voltage) for decreasing their throughput as close as possible to the minimum (found in the previous step), is calculated using Equations 5 and 4 respectively (lines 8-9).

Algorithm 1: Algorithm for Balancing Throughput

Input : Per-core IPC (IPC_i), LDO voltage levels (LDO^j_{out}), Variability modeling parameters (V^i_{th}, L^i_{eff})

Output: Per-core $Voltage$ and $Frequency$ allocation for balancing throughput

foreach $i \in Cores$ **do**
  $Vdd_i \longleftarrow LDOout(LDO, V^i_{dd})$
  $F_i \longleftarrow f_{NTC}$
  $T_i \longleftarrow F_i \cdot IPC_i$
end
$T_{min} \longleftarrow min(T_i)$
foreach $i \in Cores$ **do**
  $F_i \longleftarrow \frac{T_{min}}{IPC_i}$
  $Vdd_i \longleftarrow f^{-1}(F_i, V^i_{th}, L^i_{eff})$
end

Assuming that the LDOs have discrete output voltage levels, we select the lowest level possible that is providing a higher value than the ideal V_{dd} calculated, ensuring in that way that the desired frequency can be reached. For our experiments we assumed that the LDOs have a 12.5 mV output voltage resolution, meaning that they can provide the desired voltage in steps of the aforementioned voltage value. As mentioned in the previous subsection (3.2), the LDOs provide an output voltage range of 100mV (0.35V - 0.45V), offering, as a consequence, 8 discrete voltage levels.

The algorithm described above sustains and guarantees the throughput exhibited from each application (in the sense that its goal is not to maximize it like in other research attempts), while minimizing the overall power consumption. It introduces a low overhead and can be run at runtime while it scales linearly with the number of cores, i.e. $\mathcal{O}(n)$ complexity. Additionally, each per-core computation (throughput calculation and voltage/frequency allocation) is orthogonal to each other and can be computed fully in parallel. In this work, it is applied to homogeneous and symmetrical workloads but it can be applied to unbalanced ones considering each time the target platform (e.g. cloud, server etc.), the application scenario (e..g single/multiple applications/instances, number of threads etc.) and the designer's goals, but this is left as future work.

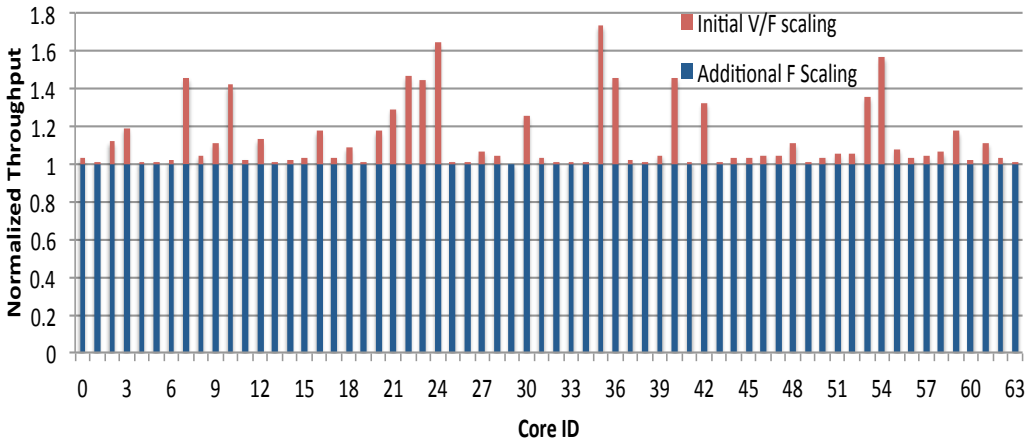


Figure 5: Per-core throughput: balancing obtained by the proposed algorithm.

5. Experimental Results

5.1. Experimental Setup

We use the Sniper manycore simulator [30] for obtaining the performance counters and McPAT [31] for the initial power estimation. Since McPat is not validated against near-threshold voltages, we use it for obtaining the ST values and then we scale accordingly in order to calculate the near threshold ones, using the models for NTC provided in [24]. In order to characterize the process variation at the NT region we deployed the Varius-NTV microarchitectural model [22]. The applications simulated are taken from the Splash-2 [32] and Parsec [33] benchmark suites, which provide high performance parallel workloads. Each application has different characteristics and exhibits a different behavior, i.e. performance, depending on the number of allocated resources, i.e.cores. The adopted on-chip voltage regulation scheme [27] enables voltage scaling response times on the order of hundreds of nanoseconds up to a few microseconds. The proposed voltage/frequency tuning framework is applied over epochs of 1ms, thus even in the case that the throughput balancing algorithm is used and triggered constantly due to regular workload imbalances, the overhead of the transition is negligible, i.e. several orders of magnitude lower than the aforementioned epoch.

Since we are simulating homogeneous multi-threaded applications, we consider a discrete number of configurations with the following number of cores: 4, 8, 16, 24, 32, 64.A summary of the experimental setup used to evaluate the

Table 1: Experimental Setup: Platform Parameters

Parameters	Value
Process Technology	22nm
STC Supply Voltage	1.05V
NTC Supply Voltage	0.45V
Nominal $V_{th}/\sigma_{V_{th}}$	0.23V/0.025
Number of Cores/Core Area	64/$6mm^2$
Tile/	4cores
Private Cache Size/Area	320KB/$4.14mm^2$
Last Level Cache Size – Area	8 MB / $15.52mm^2$

,

methodology is presented in Table 1. Intel's Nehalem architecture (22nm technology node) has been adopted for determining the core and cache types and sizes. The maximum number of cores is 64, divided in tiles consisting of 4 cores and a shared last level cache (LL$) of 8MB and each core owns a private instruction and data cache (P$).

5.2. Energy Savings: NTC vs STC

Figure 6 shows the energy savings obtained when going from 4 cores at ST voltage to 64 cores at NT voltage for each benchmark, assuming the power delivery system described in 3.2. All the applications are benefited significantly by running in NTC mode, i.e the minimum energy reduction is 55%. However, there is a large variation in the saving between different applications. This is caused from the distinct workload characteristics and the inherent parallelism of each benchmark, i.e. the performance scalability w.r.t the increase of the number of cores (4 to 64), since it impacts the minimum frequency to be sustained and thus the minimum V_{dd} to be deployed. As depicted in Figure 6, a maximum of 76% decrease in energy is observed for the Raytrace application, whereas the minimum gain is around 55% for the Cholesky application. The average savings are 66%.

5.3. Throughput Balancing

Figure 7, depicts the per-core throughput, normalized to the minimum value, after running the algorithm proposed in the previous section. In order to evaluate its efficiency we define a measure similar to standard deviation (σ) by substituting the mean with the minimum. Its purpose is to measure the distance of the throughput values from the minimum for each core. This is because our goal is

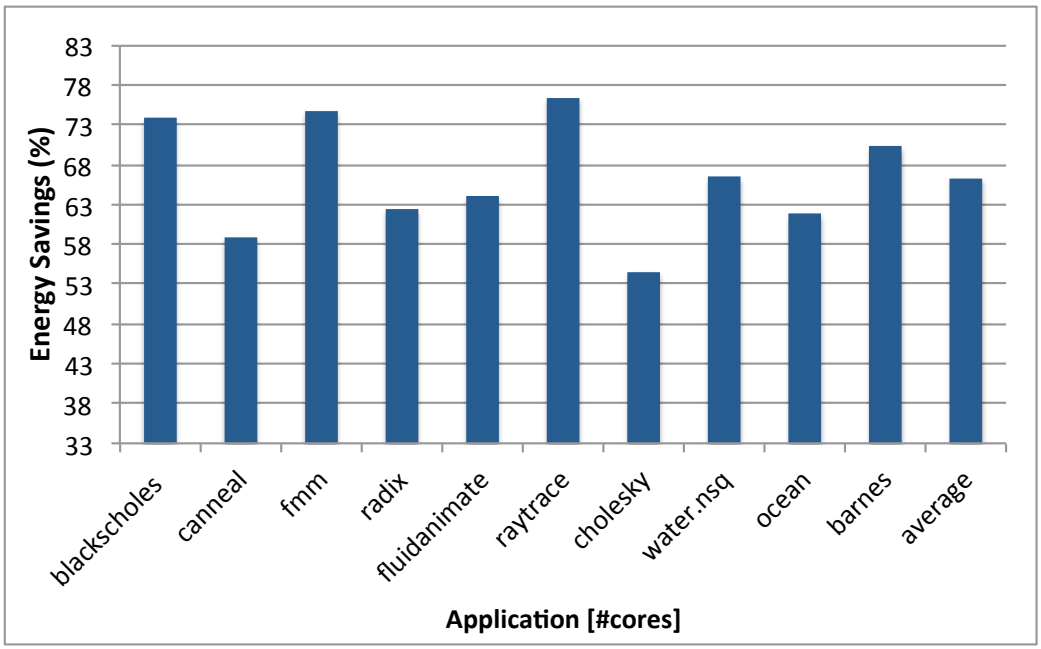


Figure 6: Energy savings: 4-core STC chip versus 64-core NTC

to balance per-core throughput as much as possible w.r.t. the core with the lowest throughput. We define the following measure:

$$\zeta = \sqrt{\sum_{i} \frac{(T_i - T_{min})^2}{\#cores}} \tag{6}$$

Whereas the throughput ζ-deviation, as defined in Equation 6, for the 16-threaded Cholesky application is 77.8% before balancing, it drops to 10.5% after applying our algorithm. For 64 cores, the same measure is increased to 95.8% before balancing but it is reduced down to 30% after applying our technique (69% reduction). As we can see in Figure 7, most of the cores exhibit a similar throughput, except for some of them, represented by the peaks in red color in the graph, e.g. 24, 35, 54, that are much faster and with the specific voltage range provided by the regulators it is impossible to reduce their throughput any further. For each application the ζ-deviation is reduced by 70% on average. The previous results are obtained assuming that, given the allocated voltage, the maximum achievable frequency is assigned to each core. As mentioned before, throughput in Figure 7 is normalized w.r.t. minimum value, therefore, the faster cores can exhibit a quite higher throughput (represented as red colored peaks in the graph) due to the higher

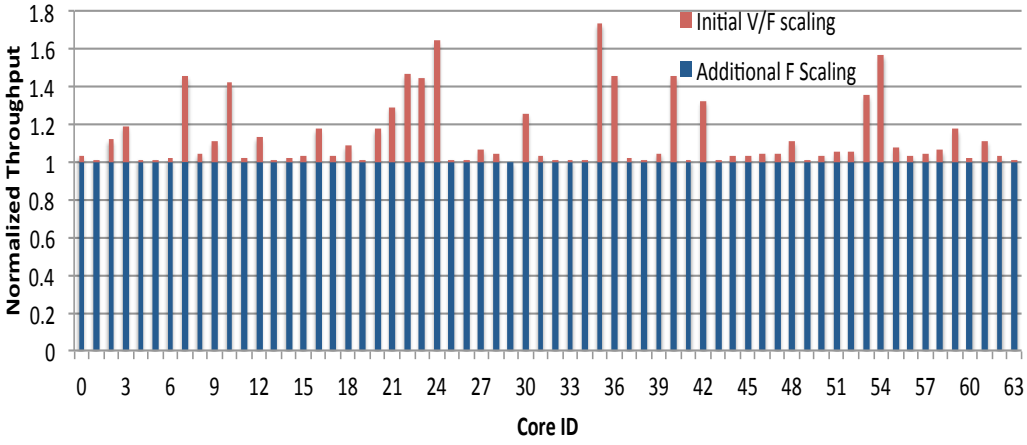


Figure 7: Per-core throughput: balancing obtained by the proposed algorithm.

frequency that can be assigned to them. As an extra, fine tuning optimization, we can slow down those cores, by reducing the frequency in order to perfectly balance the throughput and eliminate the variation completely, obtaining some extra power savings (reported in section 5.5).

The technique proposed, may look general and not specifically customized for NTC. However, the important observation here is that 1) the effect of variability is greatly increased in NTC causing a much greater imbalance than in STC and 2) the number of resources (cores) is much larger in NTC, creating the potential of greater power/energy savings. Specifically, when assuming a 4-core platform in STC the average savings for all applications is just just 16%, whereas even for a 16-core platform the average savings don't exceed 19%.

5.4. Quantitative comparison with state-of-art

Having described our methodologies for sustaining performance and balancing throughput in NTC, as well as the results obtained in each case, we take a step further to see how they are connected and how they compare to each other.

If we consider a single voltage scenario (similar to the approach presented in [3]) , then the only degree of freedom we have is frequency scaling. This is also mentioned as Single Voltage/Multiple Frequencies (SVMF) approach. Assuming a performance sustainability scenario, we compare this approach to the one presented in Section 4. We also consider that the voltage provided to the SVMF platform is the maximum one delivered by the power delivery architecture used in

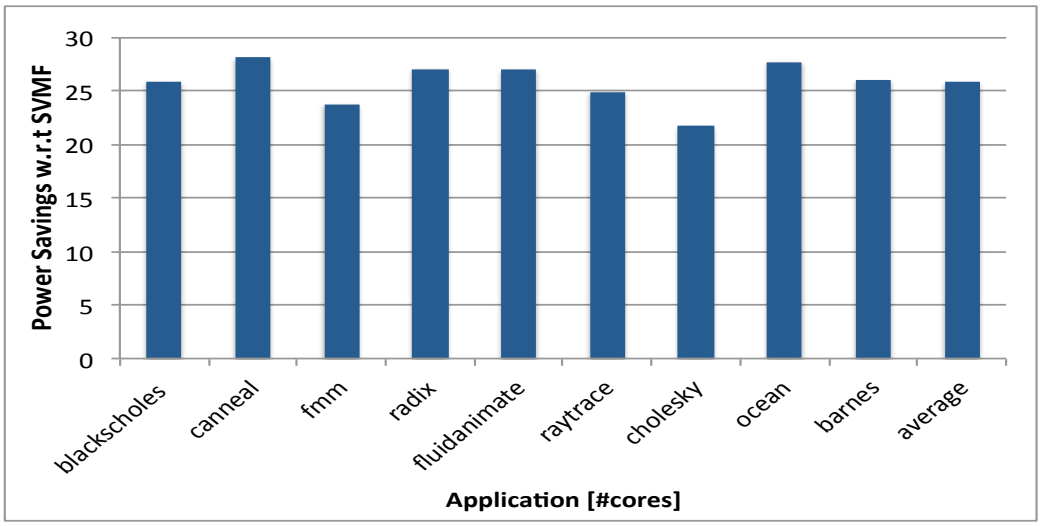


Figure 8: Power savings obtained when compared with the SVMF approach.

this work. Figure 8 depicts the power savings we get when we compare the 2 approaches. The savings we get are more or less similar for all applications with an average value of 25%. This is because, even though the frequency can be scaled down to the one needed for sustaining the performance requirement, the voltage cannot be scaled accordingly, wasting extra power when not needed.

Next, we move on to comparing 3 different approaches as well as evaluating the impact of throughput balancing discussed in the previous section. First, we evaluate a power management technique similar to the one described above (SVMF) [3]: This time we allow the voltage to be adjustable, but after calculating (according to Section 4) the frequency that satisfies the required constraint, the minimum voltage provided by the power delivery system that guarantees the aforementioned frequency, while taking into account process variation, is allocated to the whole platform, i.e. the slowest core is determining the voltage. Then a Multiple-Voltages/Multiple-Frequencies (MVMF) power management technique is deployed, taking advantage of the multiple voltage levels and the fine-grained, per-core voltage scaling to reduce the per-core variability and maximum per-core frequency. Last, the methodology presented in this paper, Multiple-Voltages/Single-Frequency (MVSF) combined with the throughput balancing heuristic is used. In that way we can sustain performance, mitigate variability and reduce energy consumption, showing that this technique is very efficient when dealing with this kind of applications.

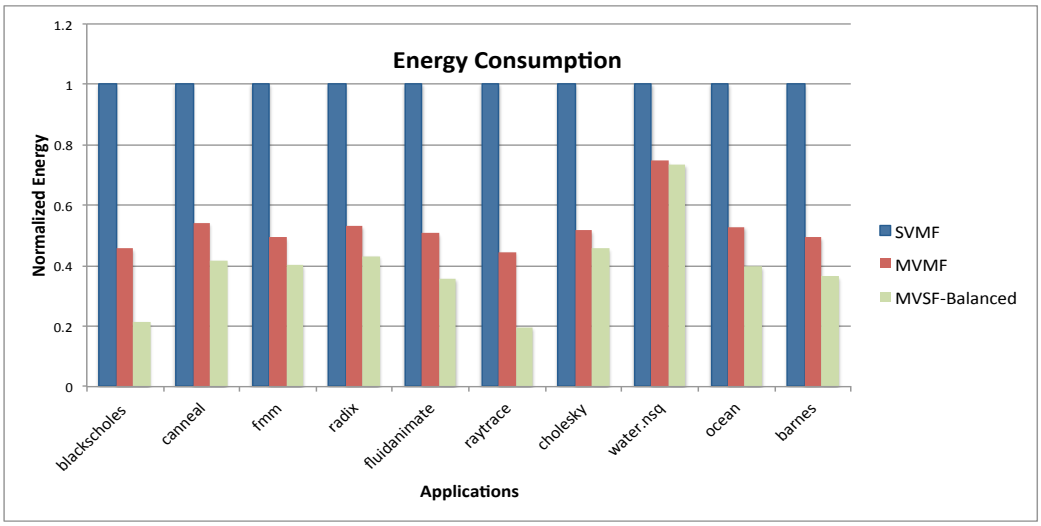


Figure 9: Energy consumption comparison of the SVMF, MVMF and MVSF (w.throughput-balancing) techniques.

The results are depicted in Figure 9. As we can see, providing an adjustable per -core voltage decreases dramatically the energy consumption, 50% on average (ranging from 35% up to 55% depending on the scalability of each application. However, when throughput balancing is taken into account, we can reduce energy even further by just using one per-chip frequency, i.e. the one that meets the performance constraint. As we can see, the results vary depending again on the workload characteristics. For example, *Blackscholes*, which is an embarrassingly parallel application, can reduce the energy consumption by 50%, whereas *Water.nsq* does not have essentially any benefit. On average, the technique delivers an extra energy reduction of 26%.

Finally, we take the previous analysis one step further and we compare the SVMF with the MVSF approach after balancing throughput while running at the minimum frequency for sustaining the ST performance. Additionally, we compare these two techniques with a best effort throughput balancing technique: we try to run our applications as fast as the resources provided can allow us, relaxing the performance constraint, but still applying the balancing throughput algorithm described in Section 4.2

Figure 10 shows that the MVSF technique provides better results compared to the SVMF one. This can be easily explained by the fact that while in the first case after calculating the platform frequency, we can tune in a fine-grained, per-

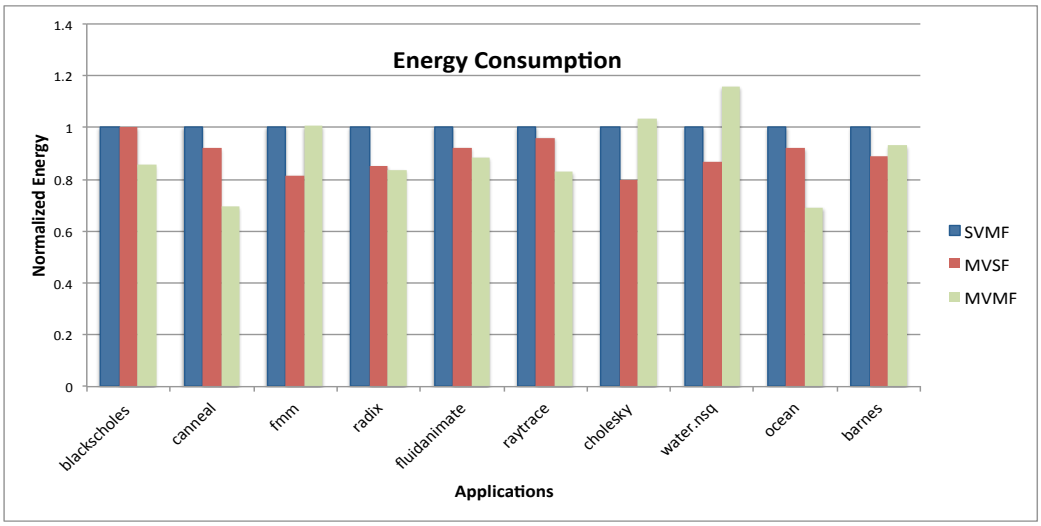


Figure 10: Energy consumption comparison of the different techniques.

core manner the voltage, allowing the faster cores to operate under a lower V_{dd}, in the SVMF approach the common voltage (which is the one enabling the slowest core to run in the required frequency for sustaining performance), increases the power consumption and eliminates the potential voltage scaling-induced power savings delivered from the faster cores. As we can observe, it provides always better results in terms of energy consumption, 35% on average. The least benefit is observed for the *Water.nsq* application, which scales poorly with the number of cores, not allowing in that way great savings because of the increased frequency that is needed in order to sustain its ST performance.

The results regarding the third technique (MVMF) are also interesting. The throughput balancing technique, can provide significant energy reduction, up to 45% and 17% compared to the SVMF and MVSF techniques (*Canneal, Ocean*), but not in all cases. For example, the energy consumed in the *Water.nsq* benchmark is 15% higher than in the SVMF approach and in some other benchmarks like *Radix* and *Barnes* we have similar energy consumption with the MVSF approach. This initally strange behavior can be explained by the fact that in this scenario we deploy a best effort approach, not considering performance sustainability, but going even faster when possible. This can be an interesting trade-off, especially when we have an optimization problem, like maximizing performance given an energy constraint. In the case of *Water.nsq*, the speedup is not big enough, so we end up consuming more energy. The speedup of the various applications is

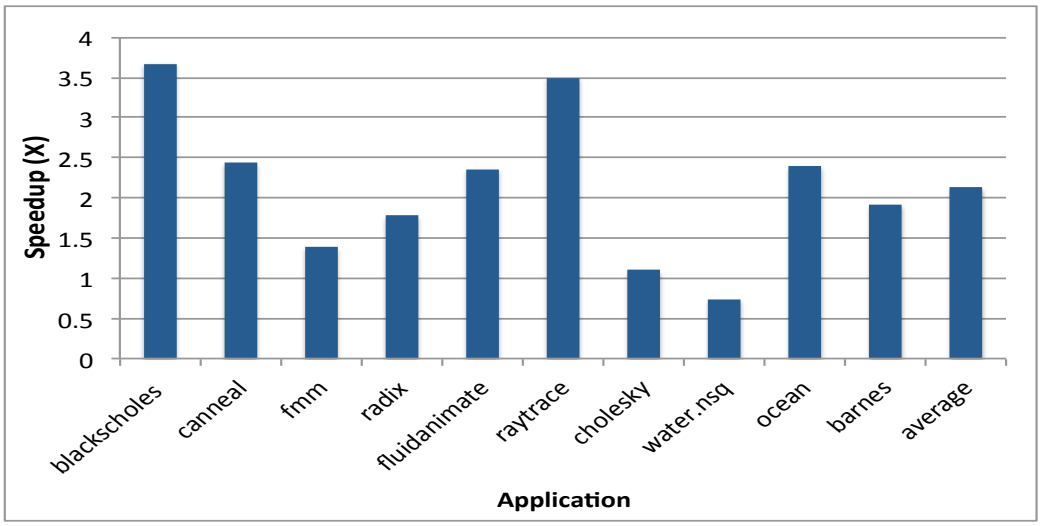


Figure 11: Speedup of the MVMF technique

shown in Figure 11. Some benchmarks (*Blackscholes*, *raytrace*) get a significant speedup (3x), while the average value is 2x. The most important point, that we need to stress out is, that by using the throughput balancing algorithm proposed in this work, we can have energy savings even while improving performance compared to the first two techniques (SVMF, MVSF).

5.5. Impact of Resource Allocation on NTC Energy Efficiency

Till now, we considered that granted a platform that consists of a specific number of cores (in our case 64), we allocate all of the resources each time when we want to run in NTC. However, this may not be the case if we just want to run in the most energy efficient configuration and we do not care about meeting a specific performance constraint, e.g. STC performance sustainability. In this section, for each application, we characterize each configuration in terms of energy efficiency and we find the most efficient one. Energy efficiency, as shown in Equation 7, is defined as Throughput over Power and is measured in Instructions/Joule or MIPS/Watt equivalently. After running our algorithm, we can determine the most energy efficient configuration for each application.

$$\textit{Energy Efficiency} = \frac{Throughput}{Power} \tag{7}$$

Table 2 summarizes the results. Applications have different optimal configuration points (#cores) depending on their workload characteristics, i.e. after a cer-

tain point there is no sense in allocating more resources because the performance-throughput is not increased significantly in order to compensate for the increased power consumed by the extra cores. For each optimal configuration, we calculate the power savings achieved by applying our optimization algorithm. The power savings are quite high, 39% on average, (varying from 34% up to 44%) and they become even more significant when considering that the overall performance was not impacted. As mentioned in the previous subsection, by performing an extra frequency scaling for the faster cores, we can eliminate the throughput variation completely, increasing ideally the average power savings from 39% to 43.5% (varying from 37% up to 51%).

Table 2: Energy Efficient Optimal Configuration

App	#cores	App	#cores
blackscholes	48	canneal	48
fmm	32	ocean	32
fluidanimate	24	raytrace	24
streamcluster	48	cholesky	64
swaptions	8	water.nsq	8
radix	8	barnes	8

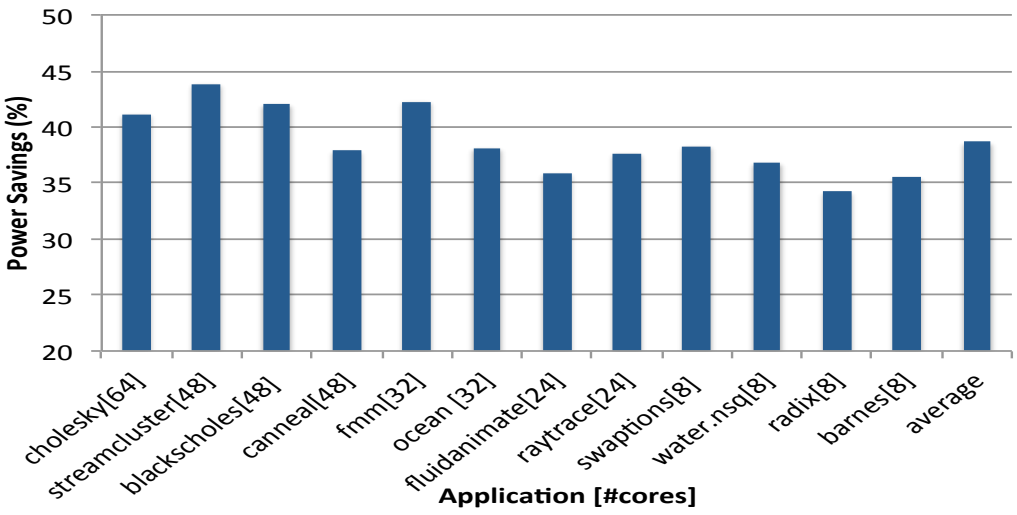


Figure 12: Power savings obtained by the proposed technique.

6. Conclusion

In this work, we adressed the problem of energy-efficient performance sustainability in NTC manycores under workload- and process-variability. We investigated the benefits of NTC for highly parallel applications, while considering the power inefficiencies of a scalable and low overhead NT power delivery network. We showed that NTC can provide performance guarantees while delivering significant energy savings (66% on average). Finally, we adressed throughput imbalance caused by the excessive impact of process variability at near-threshold (NT) voltages, by proposing a runtime algorithm that manages to reduce both the variation of the per-core throughput w.r.t. the minimum and the power consumption, on average, by 70% and 43.5% respectively for the reported results, while not impacting the overall performance.

References

[1] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, T. N. Mudge, Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits, Proceedings of the IEEE 98 (2010) 253–266.

[2] N. Pinckney, K. Sewell, R. G. Dreslinski, D. Fick, T. Mudge, D. Sylvester, D. Blaauw, Assessing the performance limits of parallelized near-threshold computing, in: Proceedings of the 49th Annual Design Automation Conference, ACM, pp. 1147–1152.

[3] U. R. Karpuzcu, A. Sinkar, N. S. Kim, J. Torrellas, Energysmart: Toward energy-efficient manycores for near-threshold computing, in: High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on, IEEE, pp. 542–553.

[4] I. Stamelakos, S. Xydis, G. Palermo, C. Silvano, Variation-aware voltage island formation for power efficient near-threshold manycore architectures, in: Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific, IEEE, pp. 304–310.

[5] C. Silvano, G. Palermo, S. Xydis, I. Stamelakos, Voltage island management in near threshold manycore architectures to mitigate dark silicon, in: Proceedings of the conference on Design, Automation & Test in Europe, European Design and Automation Association, p. 201.

[6] A. A. Sinkar, H. Wang, N. S. Kim, Maximizing throughput of power/thermal-constrained processors by balancing power consumption of cores, in: Quality Electronic Design (ISQED), 2014 15th International Symposium on, IEEE, pp. 633–638.

[7] Y. Turakhia, G. Liu, S. Garg, D. Marculescu, Thread progress equalization: Dynamically adaptive power and performance optimization of multi-threaded applications, CoRR abs/1603.06346 (2016).

[8] D. Fick, R. G. Dreslinski, B. Giridhar, G. Kim, S. Seo, M. Fojtik, S. Satpathy, Y. Lee, D. Kim, N. Liu, et al., Centip3de: A 3930dmips/w configurable near-threshold 3d stacked system with 64 arm cortex-m3 cores, in: Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International, IEEE, pp. 190–192.

[9] S. Khare, S. Jain, Prospects of near-threshold voltage design for green computing, in: 2013 26th International Conference on VLSI Design and 2013 12th International Conference on Embedded Systems, IEEE, pp. 120–124.

[10] J. Wang, J. Zhang, W. Zhang, K. Qiu, T. Li, M. Wu, Near threshold cloud processors for dark silicon mitigation: the impact on emerging scale-out workloads, in: Proceedings of the 12th ACM International Conference on Computing Frontiers, ACM, p. 4.

[11] A. Pahlevan, J. Picorel, A. P. Zarandi, D. Rossi, M. Zapater, A. Bartolini, P. G. Del Valle, D. Atienza, L. Benini, B. Falsafi, Towards near-threshold server processors, in: 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, pp. 7–12.

[12] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, S. Borkar, Near-threshold voltage (ntv) design: opportunities and challenges, in: Proceedings of the 49th Annual Design Automation Conference, ACM, pp. 1153–1158.

[13] D.-C. Juan, S. Garg, J. Park, D. Marculescu, Learning the optimal operating point for many-core systems with extended range voltage/frequency scaling, in: Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2013 International Conference on, IEEE, pp. 1–10.

[14] W.-C. Hsieh, W. Hwang, All digital linear voltage regulator for super-to near-threshold operation, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 20 (2012) 989–1001.

[15] S. K. Lee, D. Brooks, G.-Y. Wei, Evaluation of voltage stacking for near-threshold multicore computing, in: Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design, ACM, pp. 373–378.

[16] X. He, G. Yan, Y. Han, X. Li, Superrange: wide operational range power delivery design for both stv and ntv computing, in: Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, IEEE, pp. 1–6.

[17] T. N. Miller, X. Pan, R. Thomas, N. Sedaghati, R. Teodorescu, Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips, in: High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on, IEEE, pp. 1–12.

[18] I. Stamelakos, A. Khajeh, A. Eltawil, G. Palermo, C. Silvano, F. Kurdahi, A system-level exploration of power delivery architectures for near-threshold manycores considering performance constraints, in: VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on, IEEE, pp. 484–489.

[19] W. Lee, Y. Wang, M. Pedram, Vrcon: dynamic reconfiguration of voltage regulators in a multicore platform, in: Proceedings of the conference on Design, Automation & Test in Europe, European Design and Automation Association, p. 365.

[20] A. A. Sinkar, H. R. Ghasemi, M. J. Schulte, U. R. Karpuzcu, N. S. Kim, Low-cost per-core voltage domain support for power-constrained high-performance processors, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 22 (2014) 747–758.

[21] D. Kanter, Inside Nehalem: Intel's future processor and system. http://www.realworldtech.com, 2008.

[22] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, J. Torrellas, VARIUS-NTV: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages, in: IEEE/IFIP International Conference on Dependable Systems and Networks, DSN, pp. 1–11.

[23] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, J. Torrellas, Varius: A model of process variation and resulting timing errors for microarchitects, Semiconductor Manufacturing, IEEE Transactions on 21 (2008) 3–13.

[24] D. Marković, C. C. Wang, L. P. Alarcon, T.-T. Liu, J. M. Rabaey, Ultralow-power design in near-threshold region, Proceedings of the IEEE 98 (2010) 237–252.

[25] G. G. Faust, R. Zhang, K. Skadron, M. R. Stan, B. H. Meyer, Archfp: Rapid prototyping of pre-rtl floorplans, in: VLSI-SoC, pp. 183–188.

[26] B. H. Calhoun, A. Wang, N. Verma, A. Chandrakasan, Sub-threshold design: The challenges of minimizing circuit energy, in: Proceedings of the 2006 International Symposium on Low Power Electronics and Design, ISLPED '06, ACM, New York, NY, USA, 2006, pp. 366–368.

[27] Y. Okuma, K. Ishida, Y. Ryu, X. Zhang, P.-H. Chen, K. Watanabe, M. Takamiya, T. Sakurai, 0.5-v input digital ldo with 98.7% current efficiency and 2.7-μa quiescent current in 65nm cmos, in: Custom Integrated Circuits Conference (CICC), 2010 IEEE, IEEE, pp. 1–4.

[28] N. Pinckney, K. Sewell, R. G. Dreslinski, D. Fick, T. Mudge, D. Sylvester, D. Blaauw, Assessing the performance limits of parallelized near-threshold computing, in: Proceedings of the 49th Design Automation Conference, pp. 1147–1152.

[29] S. Dighe, S. Vangal, P. Aseron, S. Kumar, T. Jacob, K. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, et al., Within-die variation-aware dynamic-voltage-frequency scaling core mapping and thread hopping for an 80-core processor, in: Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International, IEEE, pp. 174–175.

[30] T. E. Carlson, W. Heirman, L. Eeckhout, Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations, in: International Conference for High Performance Computing, Networking, Storage and Analysis (SC).

[31] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, N. P. Jouppi, McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures, in: Proceedings of the 42nd Annual

IEEE/ACM International Symposium on Microarchitecture, MICRO 42, pp. 469–480.

[32] J. M. Arnold, D. A. Buell, E. G. Davis, Splash 2, in: Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures, ACM, pp. 316–322.

[33] C. Bienia, S. Kumar, J. P. Singh, K. Li, The parsec benchmark suite: Characterization and architectural implications, in: Proceedings of the 17th international conference on Parallel architectures and compilation techniques, ACM, pp. 72–81.