NOVEL MANY-CORE ARCHITECTURES FOR ENERGY-EFFICIENCY

BY

ULYA R. KARPUZCU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Josep Torrellas, Chair
Professor Wen-Mei Hwu
Professor Sanjay Patel
Professor Naresh Shanbhag
Assistant Professor Nam Sung Kim
Chris Wilkerson, Intel Corporation

ABSTRACT

Ideal CMOS device scaling relies on scaling voltages down with lithographic dimensions at every technology generation. This gives rise to faster circuits due to increased frequency and smaller silicon area for the same functionality. In this model, the dynamic power density (dynamic power per unit area) stays constant.

In recent generations, however, to keep leakage current under control, the decrease in the transistor's threshold voltage has stopped. As a result, already at the current technology generation, processor chips can include more cores and accelerators than can be active at any given time - and the situation is getting worse. This effect, broadly called the *Power Wall*, presents a fundamental challenge that is transforming the many-core architecture landscape.

This thesis focuses on addressing the Power Wall problem in two novel and promising ways: By (1) managing and trading-off the processor aging (or wear-out) rate for energy efficiency (the *BubbleWrap* many-core), and (2) exploring near-threshold voltage operation (the *Polyomino* many-core).

BubbleWrap assumes as many cores on chip as the Moore's Law suggests, and exploits the resulting redundancy – as not all of the cores can be powered on simultaneously – to extract maximum performance by trading off power and service life on a per-core basis. To achieve this, BubbleWrap continuously tunes the supply voltage applied to a core in the course of its service life (but not the frequency), exploiting any aging guard-band instantaneously left in the core. This renders one of the following regimes of operation: Consume the least power for the same performance and processor service life; attain the highest performance for the same service life while respecting the given power constraint; or attain even higher performance for a shorter service life while respecting the given power constraint. Effectively, BubbleWrap runs the core at a better operating point by aggressively using up, at all times, all the aging-induced guard-band that the designers have included - preventing any waste of it, unlike in all current processors.

One way to attain energy-efficient execution is to reduce the supply voltage to a value only slightly higher than a transistor's threshold voltage. This regime is called Near-Threshold Voltage

(NTV) computing, as opposed to the conventional Super-Threshold Voltage (STV) computing. One drawback of NTV is the higher magnitude of parameter variations, namely the deviation of device parameters from their nominal values. To cope with process variations in present and future NTV designs, this thesis first builds on an existing model of variations at STV and develops the first architectural model of process variations at NTV. Secondly, using the model, this study shows that supporting multiple on-chip voltage domains to cope with process variations will not be cost-effective in near-future NTV designs.With this insight, this thesis introduces Polyomino a simple many-core architecture which can effectively cope with variations at NTV.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 MOTIVATION

Ideal CMOS device scaling relies on scaling voltages down with lithographic dimensions at every technology generation. This gives rise to faster circuits due to increased frequency and smaller silicon area for the same functionality. In this model, the dynamic power per unit area stays constant. This is because the energy per switching event decreases enough to compensate for the increased energy due to having more devices in the same area and switching them faster.

In recent generations, however, to keep leakage current under control, the decrease in the transistor's threshold voltage has stopped. This, in turn, has prevented the supply voltage from scaling. The net result is that the compensation effect does not exist any more. As more transistors are integrated on a fixed-sized chip at every generation, the chip power increases rapidly. If the chip power budget is fixed due to system cooling constraints and the associated costs, a growing gap emerges between what can be placed on a chip and what can be powered-on simultaneously. As a result, already at the current technology generation, processor chips can include more cores and accelerators than can be active at any given time and the situation is getting worse. This effect, broadly called the *Power Wall*, presents a fundamental challenge that is transforming the many-core architecture landscape.

This thesis focuses on addressing the Power Wall problem in two novel and promising ways: By (1) managing and trading-off the processor aging (or wear-out) rate for energy efficiency (the *BubbleWrap* many-core), and (2) exploring near-threshold voltage operation (the *Polyomino* many-core).

The BubbleWrap many-core is an architecture that manages the rate of processor aging (or wear-out) and trades it off for performance or energy efficiency. BubbleWrap introduces the novel concept of *Dynamic Voltage Scaling for Aging Management* (DVSAM). The idea is to continuously tune the supply voltage applied to a core in the course of its service life (but not the frequency), exploiting any aging guard-band instantaneously left in the core. The goal can be one of the following regimes of operation: Consume the least power for the same performance and processor service life; attain the highest performance for the same service life while respecting a given power con-

straint; or attain even higher performance for a shorter service life while respecting a given power constraint. Effectively, we are running the core at a better operating point by aggressively using up, at all times, all the aging-induced guard-band that the designers have included - preventing any waste of it, unlike in all current processors.

BubbleWrap takes its name from the third environment above. In this case, we identify the set of most power-efficient cores in a variation-affected die - the largest set that can be simultaneously powered-on. These cores are reserved as *Throughput* cores dedicated to parallel section execution. The rest of the cores are designated as *Expendable* and are dedicated to accelerating sequential sections. Expendable cores are sacrificed one at a time, by running each of them at an elevated supply voltage for a short, few-month-long service life, until the core completely wears-out and is discarded figuratively, as if popping bubbles in a bubble wrap that protects Throughput cores. As a result, BubbleWrap provides substantial performance gains over a plain chip.

One way to attain energy-efficient execution is to reduce the supply voltage to a value only slightly higher than a transistor's threshold voltage. This regime is called Near-Threshold Voltage (NTV) computing, as opposed to the conventional Super-Threshold Voltage (STV) computing. One drawback of NTV is a degradation in core frequency, which may be tolerable through more parallelism in the application - i.e., using more on-chip cores. Unfortunately, a more important problem at NTV is the higher magnitude of parameter variations, namely the deviation of device parameters from their nominal values.

Already at STV, process variations result in substantial power and performance losses in many-cores. At NTV, the same amount of process variations causes even larger changes in speed and power across the transistors in a chip. Effectively coping with process variations at NTV at the architecture level is difficult for at least two reasons. First, all of the existing architectural models of process variations apply only to STV rather than NTV. Second, conventional techniques to address process variations at STV typically rely on supply voltage tuning (in the form of multiple on-chip voltage domains with voltage scaling), which is power inefficient.

To cope with process variations in present and future NTV designs, this thesis first builds on an existing model of variations at STV and develops the first architectural model of process variations at NTV. Secondly, using the model, this study shows that supporting multiple on-chip voltage domains to cope with process variations will not be cost-effective in near-future NTV designs. There are three reasons for it: the on-chip voltage regulators power losses, the increased supply voltage

guardband needed to handle deeper voltage droops (due to lower capacitance per domain), and the practical fact that a domain still includes many cores. With this insight, this thesis introduces Polyomino, a simple many-core architecture which can effectively cope with variations at NTV. Polyomino eschews multiple voltage domains and relies on fine-grain frequency domains to optimize execution under variations. Thanks to Polyomino's simplicity, a scheduler that knows the chips variation profile can effectively assign core clusters to jobs. In this manner, Polyomino runs workloads significantly more energy efficiently than conventional architectures.

2 TRADING-OFF THE AGING RATE FOR ENERGY EFFICIENCY: THE BUBBLEWRAP MANY-CORE

2.1 Introduction

Ideal CMOS device scaling [1] relies on scaling voltages down with lithographic dimensions at every technology generation — giving rise to faster circuits due to increased frequency and smaller silicon area for the same functionality. In this model, the dynamic power per unit area stays constant. This is because the energy per switching event decreases enough to compensate for the increased energy due to having more devices in the same area and switching them faster.

In recent generations, however, to keep leakage current under control, the decrease in the transistor's threshold voltage (Vth) has stopped. This, in turn, has prevented the supply voltage (Vdd) from scaling [2,3]. Given the quadratic dependence of the dynamic energy on supply voltage, the net result is that the compensation effect explained above does not exist any more. As more transistors are integrated on a fixed-sized chip at every generation, the chip power increases rapidly. Chip power does not scale anymore.

If we fix the chip power budget due to system cooling constraints and the associated costs, we easily realize that there is a growing gap between what can be placed on a chip and what can be powered-on simultaneously. For example, Figure 2.1 shows data computed from the ITRS 2008 update [4] assuming Intel Core i7-like [5] cores and a constant 100W chip power budget. The figure compares the number of cores that can be placed on a chip at a given year (normalized to year 2008 numbers) and the number of those that can be powered-on simultaneously. The growing gap between the two curves shows the *Many-Core Power Wall*. Soon, many cores may have to remain powered-off.

Another key effect of aggressive scaling is increasing parameter variation [6]. Variation can manifest as static, spatial variations across the chip, or dynamic, temporal variations as a processor is used. A significant contributor to the latter is device wearout or aging. Aging induces a progressive slowdown in the logic as it is being used.

Recently, processor aging has been the subject of much work (e.g., [7–14]). It is well accepted

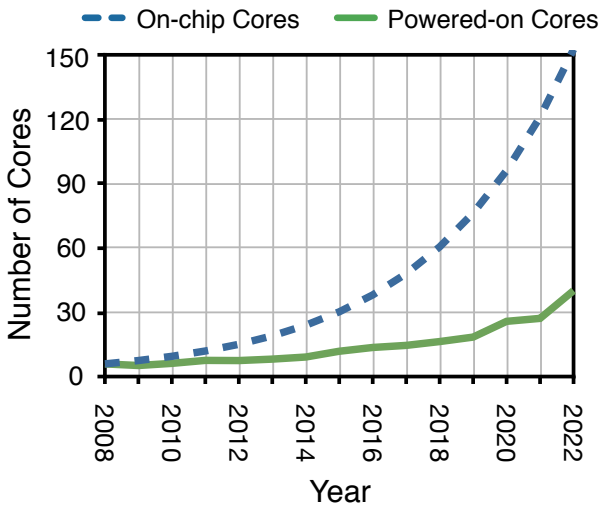


Figure 2.1: The many-core power wall, based on data from ITRS projections.

that the aging rate is highly impacted by Vdd and temperature (T), where higher values increase the aging rate. Consequently, approaches to slow-down aging by operating at lower Vdd or T have been introduced [14]. We observe that such approaches to change the aging rate typically affect performance and power. Consequently, they could help push back the many-core power wall.

Based on this observation, we propose a novel scheme for managing processor aging that attains higher performance or lower power consumption. We call our scheme *Dynamic Voltage Scaling for Aging Management (DVSAM)*. The idea is to continuously tune Vdd (but not the frequency), exploiting any currently-left aging guard-band. The goal can be one of the following: consume the least power for the same performance and processor service life; attain the highest performance for the same service life and within power constraints; or attain even higher performance for a shorter service life and within power constraints.

We also propose *BubbleWrap*, a novel many-core architecture that makes extensive use of DVSAM to push back the many-core power wall. BubbleWrap identifies the most power-efficient set of cores in a variation-affected die — the largest set that can be simultaneously powered-on. It designates them as *Throughput* cores dedicated to parallel-section execution. The rest of the cores are designated as *Expendable* and are dedicated to accelerating sequential sections. BubbleWrap attains maximum sequential acceleration by sacrificing Expendable cores one at a time, running them at elevated Vdd for a significantly shorter service life each, until they completely wear-out and are discarded — figuratively, as if popping bubbles in bubble wrap that protects Throughput

$$\Delta Vth_{BTI} = \Delta Vth_{STRESS} \times (1 - \sqrt{\eta \times \frac{t_{RECOVERY}}{t_{RECOVERY} + t_{STRESS}}}) \quad \text{, where}$$
$$\Delta Vth_{STRESS} = A_{BTI} \times \left[\frac{q^3}{Cox^2} \times (Vdd - Vth_{NOM}) \times exp\left(-\frac{Ea}{2kT} + \frac{Vdd - Vth_{NOM}}{tox \times 0.5Eo}\right)\right]^{2a} \times (t_{STRESS})^a \quad (2.1)$$

cores.

In simulated 32-core chips, BubbleWrap provides substantial gains over a plain chip. For example, on average, one design runs fully-sequential applications at a 16% higher frequency, and fully-parallel ones with a 30% higher throughput.

Overall, we make two main contributions: (1) We introduce *Dynamic Voltage Scaling for Aging Management (DVSAM)*, a new scheme for managing processor aging to attain higher performance or lower power consumption. (2) We present the *BubbleWrap* many-core, a novel architecture that makes use of DVSAM to push back the many-core power wall.

In the following, Section 2.2 provides a background; Section 2.3 introduces DVSAM; Section 2.4 presents the BubbleWrap many-core; Sections 2.5 and 2.6 evaluate BubbleWrap and Section 2.7 discusses related work.

2.2 Background

2.2.1 Modeling Aging

Our analysis focuses on aging induced by Bias Temperature Instability (BTI), which causes transistors to become slower in the course of their normal use. BTI-induced degradation leads to increases in the threshold voltage (Vth) of the form $\Delta Vth_{BTI} \propto t^a$, where t is time and a is a time-slope constant. Constant a is strongly related to process characteristics, and generally takes a value between 0 and 0.5 for recent process generations [15–17].

To model ΔVth_{BTI}, we adopt the framework of Wang *et al* [17]. Vth only increases when the voltage between the gate and the source of the transistor is set to a given logic value, and decreases more slowly when the voltage is set to the opposite logic value. These conditions are called Stress and Recovery conditions, respectively. Equation 2.1 above shows ΔVth_{BTI} as a function of the time the transistor is under stress (t_{STRESS}) and recovery ($t_{RECOVERY}$). In the equation, A_{BTI}, a, Eo and η are model fitting parameters. Importantly, the equation shows that ΔVth_{BTI} depends

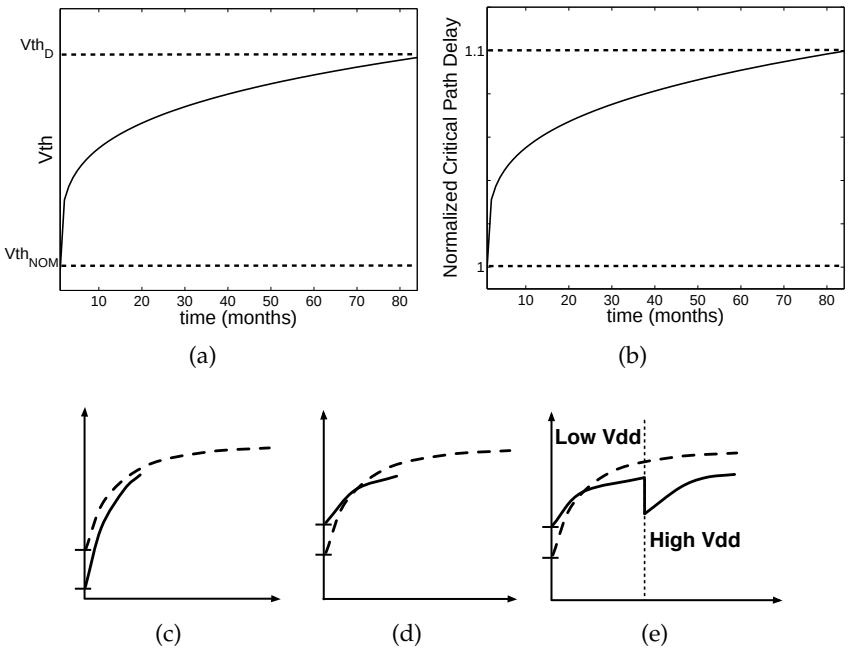


Figure 2.2: Effects of aging on Vth degradation (a) and on critical path delay degradation (b)-(e).

exponentially on the supply voltage (Vdd) and the temperature (T). Therefore, high values of Vdd or T will substantially increase the aging rate.

With this effect, Vth at a given Vdd and T is given by Equation 2.2, where Vth_{NOM}, Vdd_{NOM}, and T_{NOM} are the nominal values of these parameters, and k_{DIBL} and k_T are constants.

$$Vth = Vth_{NOM} + k_{DIBL} \times (Vdd - Vdd_{NOM}) + k_T \times (T - T_{NOM}) + \Delta Vth_{BTI} \tag{2.2}$$

The increase in Vth translates into an increase in transistor switching delay (τ) as per the alpha-power law [18] (Equation 2.3). The result is a slowdown in the processor's critical paths and, hence, its operating frequency. In the formula, $\alpha > 1$ and $\mu \propto T^{-1.5}$.

$$\tau \propto \frac{Vdd}{\mu(Vdd - Vth)^{\alpha}} \tag{2.3}$$

The aging model from Equation 2.1 was derived for devices with silicon-based dielectrics and poly gates (*Poly+SiON* devices), and verified against an industrial 65nm node [17]. To reduce gate leakage, starting from 45nm, manufacturers have introduced a new generation of devices with higher gate dielectric constants (high-k) and metal gates (*HK+MG* devices) [15, 16]. However, we argue that the model is still applicable.

To see why, we refer to a recent reliability characterization of Intel's 45nm node [19], one of the most up-to-date HK+MG processes in production. The authors report that (1) PMOS Negative BTI (NBTI) remains a major reliability concern, like it was in the predecessor Poly+SiON 65nm node, and that (2) at higher electric fields (as induced by higher Vdd), NMOS Positive BTI (PBTI) becomes significant. Further, they demonstrate that the BTI characteristics closely follow the BTI behavior of Poly+SiON devices. Specifically, the same physical phenomena cause this behavior [20].

We modify the process parameters and time-slope characteristics of the base model to reflect the new technology node in light of the observations from [20]. We assume cores like the Intel Core i7 [5], which are based on Intel's 45nm HK+MG technology. Note also that another finding from [19] is that the Hot-Carrier Injection (HCI) effect is not that significant for any realistic use condition. Hence, we focus on BTI-based aging only.

We acknowledge, however, that advances in devices in future nodes may require reconsidering the formulae. In the contemporary era of CMOS scaling, as we face more scaling limits, the pace at which new materials and features are introduced will increase [3, 21], as demonstrated by the introduction of HK+MG devices.

2.2.2 Impact of Aging

Equation 2.1 shows that ΔVth_{BTI} follows a power law with time. Since a typical value of the time slope a is between 0 and 0.5, ΔVth_{BTI} increases rapidly first and then more slowly. Let us assume a fixed ratio of stress to recovery time, and that stress and recovery periods are finely interleaved. If Vth_{NOM} is the value of Vth at the beginning of the service life, the curve $Vth = Vth_{NOM} + \Delta Vth_{BTI}$ as a function of time is shown in Figure 2.2(a). At the end of the service life, which the figure assumes is 7 years, Vth has reached Vth_D.

The switching delay of a transistor is given by Equation 2.3. As Vth increases with time due to aging, so does the switching delay τ for the same supply voltage and temperature conditions. Both PMOS and NMOS transistors suffer from BTI-induced aging — called NBTI and PBTI, respectively [19].

To determine the delay of a logic path in a processor, we follow the approach of Tiwari and Torrellas [14]. Specifically, when the path is activated, we identify the set of transistors that switch. The path delay is given by the switching delays of such transistors plus the wire delays. As

individual transistors age and their ΔVth_{BTI} increases following a power law, the total path delay can be shown to also increase following a similar curve. Specifically, the path delay increases fast at the beginning of the service life and then increases progressively more slowly.

The actual path delay increase depends on many issues, including the path composition in terms of PMOS or NMOS transistors, the amount of wire, the ratio of stress to recovery periods, the T, and the Vdd. However, the increase follows the general shape described. If we assume that the delay of the critical path of the processor increases 10% in a 7-year service life, we attain a normalized curve like the one in Figure 2.2(b). In the figure, the normalized critical path delay is 1 at the beginning, and 1.1 at the end of the service life.

Let us call the delay of the processor's critical path at the beginning of service τ_{ZG} (where ZG stands for zero guard-band). The same path will have a delay of $\tau_{NOM} = \tau_{ZG} \times (1 + G)$ at the end of the service life, where G is the timing guard-band that the path will consume during its life (10% in our example). Consequently, the processor cannot be clocked at a frequency $f_{ZG} = 1/\tau_{ZG}$ because it would soon suffer timing errors. It is clocked at the lower frequency f_{NOM} during all its service life:

$$f_{NOM} = \frac{1}{\tau_{NOM}} = \frac{1}{\tau_{ZG} \times (1 + G)} = \frac{f_{ZG}}{1 + G} \tag{2.4}$$

2.2.3 Slowing Down Aging

In recent work called Facelift [14], Tiwari and Torrellas attempt to slow down aging by perturbing the curve in Figure 2.2(b). A major knob they use is to change Vdd. Vdd affects transistor aging as per Equation 2.1 and transistor delay as per Equation 2.3. Specifically, if we increase Vdd (without changing any other parameter), transistors become faster (from Equation 2.3, since α¿1) and also age faster (from Equation 2.1). If, instead, we decrease Vdd, transistors age more slowly but they also become slower.

They then use timely Vdd changes to slow down aging. Graphically, this means forcing the curve in Figure 2.2(b) to reach a lower Y coordinate value at the end of the service life. To see the impact of timely Vdd changes, we repeat Figure 2.2(b) in Figures 2.2(c) and 2.2(d). Figure 2.2(c) shows the effect of increasing Vdd: it pushes the curve down (faster critical paths) but increases the slope of the curve (faster aging). Figure 2.2(d) shows the effect of decreasing Vdd: it pushes the curve up (slower critical paths) but reduces the slope of the curve (slower aging).

They make the observation that changing Vdd impacts (i) the aging rate and (ii) the critical path delay differently within the course of the processor service life. Specifically, it impacts the aging rate strongly (positively or negatively) at the beginning of the service life, and little toward the end. In contrast, it impacts the delay more uniformly across time. Consequently, they propose to *apply a low Vdd* toward the beginning of the service life. At that time, it reduces the aging rate the most and, therefore, slows down aging the most. Moreover, there is still substantial guard-band available to tolerate the lengthening of the critical path delay. They propose to *apply a high Vdd* toward the end of the service life. At that time, it still speeds up the critical path delay, while it increases the aging rate the least. The result of using this strategy is shown in Figure 2.2(e). At the end of the service life, the critical path delay (and therefore the wearout of the processor) is lower. Consequently, the authors can run the processor at a constant frequency throughout the service life that is higher than f_{NOM}.

Tiwari and Torrellas [14] also use this strategy to configure cores for a shorter service life. They consolidate all the aging into the shorter life and run the processor at an even higher, constant frequency throughout the short life.

2.3 DVSAM: Dynamic Voltage Scaling for Aging Management

2.3.1 Main Idea

While Tiwari and Torrellas [14] change the Vdd of a processor only once or twice in its service life, we observe that we can manage aging better if we continuously tune Vdd in small steps over the whole service life — keeping the frequency constant as these authors do. Moreover, we observe that these changes can be done not only to improve performance, but to reduce power consumption as well.

Based on these two observations, we propose *Dynamic Voltage Scaling for Aging Management (DVSAM)*. The idea is to manage the aging rate by continuously tuning Vdd (but not the frequency), exploiting any currently-left aging guard-band. DVSAM is a novel approach to trade-off processor performance, power consumption, and service life for one another.

We propose the four DVSAM modes of Table 2.1. *DVSAM-Pow* attempts to consume the minimum power for the same performance and service life. *DVSAM-Perf* tries to attain the maximum performance for the same service life and within power constraints. *DVSAM-Short* tries to attain

even higher performance for a shorter service life and within power constraints. Finally, *VSAM-Short* is the same as DVSAM-Short but without changing Vdd with time.

Mode: Goal	Vdd Values
DVSAM-Pow: Consume minimum power for the same performance and service life ($f = f_{NOM}$, $P < P_{NOM}$).	 At $t = 0$: $Vdd << Vdd_{NOM}$ At $t = S_{NOM}$: $Vdd < Vdd_{NOM}$
DVSAM-Perf: Attain maximum performance for the same service life and a given power budget ($f > f_{NOM}$, $P > P_{NOM}$).	 At $t = 0$: $Vdd < Vdd_{NOM}$ At $t = S_{NOM}$: $Vdd > Vdd_{NOM}$
DVSAM-Short: Attain even higher performance for a shorter service life and a given power budget ($f >> f_{NOM}$, $P >> P_{NOM}$).	 At $t = 0$: $Vdd > Vdd_{NOM}$ At $t = S_{SH}$: $Vdd >> Vdd_{NOM}$
VSAM-Short: Special case: Same as DVSAM-Short but no Vdd changes with time ($f >> f_{NOM}$, $P >> P_{NOM}$).	 $\forall$ t $\in [0, S_{SH}]$: $Vdd >> Vdd_{NOM}$

Table 2.1: DVSAM modes. P denotes total power consumption, with P_{NOM} corresponding to the power consumption of a core clocked at f_{NOM} under nominal operating conditions.

DVSAM operates by aggressively trying to consume, at any given time, all the aging guard-band that would be otherwise available. Consequently, the design assumes the existence of aging sensor circuits that reliably measure the guard-band available at all times [8, 9, 11, 22–25] (Section 2.4.3). Note that circuits have additional guard-bands to protect themselves against other effects such as thermal and voltage fluctuations.

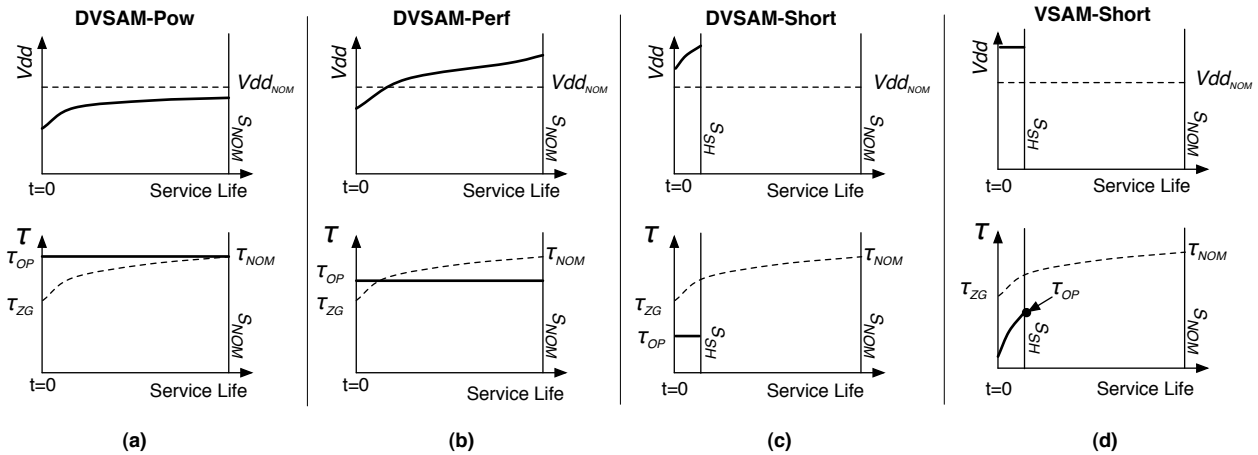


Figure 2.3: Changes in Vdd (top row) and critical path delay (bottom row) as a function of time for the different DVSAM modes.

2.3.2 Detailed Operation

To understand the DVSAM operation, note that the nominal supply voltage Vdd_{NOM} used in a processor is "over-designed" for the early parts of the processor's service life. It is designed so that, by the end of the service life, the processor's critical path is just fast enough to avert any timing error. However, earlier on in the service life, before that path aged, that path used to take less time than the cycle time — and its speed was enabled by the Vdd_{NOM}. Clearly, at that time, we could have used a lower Vdd.

This effect is seen analytically by assuming a critical path of identical gates and summing up the switching delays of all the transistors in the path using Equation 2.3. The critical path delay at the end of the service life (τ_{NOM}), when $Vth = Vth_D$, is supported by Vdd_{NOM}:

$$\tau_{NOM} \propto \frac{Vdd_{NOM}}{\mu(Vdd_{NOM} - Vth_D)^{\alpha}} \tag{2.5}$$

The same Vdd_{NOM} is used at the beginning of the service life when, because $Vth = Vth_{NOM}$, the same path only takes τ_{ZG}:

$$\tau_{ZG} \propto \frac{Vdd_{NOM}}{\mu(Vdd_{NOM} - Vth_{NOM})^{\alpha}} \tag{2.6}$$

However, since the processor is clocked at the same frequency throughout the whole service life, keeping these paths so fast is unnecessary. Consequently, DVSAM-Pow reduces Vdd in the early stages of the service life, slowing these paths but ensuring that they do not take longer than τ_{NOM}. The result is power savings. Note that the Vdd reduction becomes gradually smaller, as the paths age. It may be that, by the end of the service life, we can still use a lower Vdd than Vdd_{NOM}. The reason is that, thanks to having applied lower-than-usual Vdd to the processor over a long period, its paths have aged less than usual. Table 2.1 shows these Vdd values. In the table, the nominal service life is denoted as S_{NOM}.

Alternatively, since the paths have timing slack in the early stages of the service life, DVSAM-Perf increases the frequency of the processor, hence delivering higher performance. However, for simplicity in our design, we want to keep the elevated frequency of the processor constant over the whole service life. To do so, DVSAM-Perf also changes Vdd. Toward the early stages of the service life, to slow down aging as Tiwari and Torrellas [14], Vdd will be set slightly below Vdd_{NOM}. Toward the end of the service life, to keep up with the higher frequency of the processor, Vdd will be set above Vdd_{NOM}. Table 2.1 shows these Vdd values.

Figures 2.3(a) and 2.3(b) show the operation of the DVSAM-Pow and DVSAM-Perf modes, respectively. The top row shows the changes in Vdd as a function of time, while the bottom one shows the changes in critical path delay (τ) as a function of time. Time goes from $t = 0$ to the end of the service life S_{NOM} (e.g., 7 years). Each chart also shows, with a dotted line, the evolution of the parameter value if DVSAM was not applied. Specifically, Vdd stays constant at Vdd_{NOM} (top row), and τ changes from τ_{ZG} at $t = 0$ to τ_{NOM} at S_{NOM} — first quickly and then slowly.

Consider the Vdd charts first. As indicated above, in DVSAM-Pow, Vdd starts significantly lower than Vdd_{NOM}, gradually increases, and reaches a value below Vdd_{NOM} at S_{NOM}. In DVSAM-Perf, Vdd starts slightly lower than Vdd_{NOM}, increases faster, and ends up significantly higher than Vdd_{NOM}.

In the critical path delay charts (bottom row), both modes show a constant critical path delay (labeled τ_{OP}). This is because they dynamically tune the Vdd so that the critical path always takes exactly the same time — balancing the natural lengthening of the critical path due to aging with progressively higher Vdd. In both cases, the processor is clocked at constant frequency $f_{OP} = 1/\tau_{OP}$ over the whole service life. In DVSAM-Pow, τ_{OP} is equal to τ_{NOM}; in DVSAM-Perf, τ_{OP} is kept smaller than τ_{NOM} to enable a higher frequency. DVSAM-Perf can keep pushing τ_{OP} lower at progressively higher power costs. However, we will reach a point where constraints in Vdd, T, or service life duration will prevent any further reduction in τ_{OP}.

To attain higher performance beyond such points, we have the last two DVSAM modes. They deliver higher performance than DVSAM-Perf (at higher power) by giving up service life (Table 2.1). This means that the processor will become unusable and be discarded at a time S_{SH} (for short service life), much earlier than S_{NOM}.

Figure 2.3(c) shows the operation of DVSAM-Short. Already at the start of the service life, Vdd is set to a value higher than Vdd_{NOM} (top chart of the figure). This dramatically reduces the delay of the critical path to τ_{OP} (bottom chart), and enables the processor to cycle at a high frequency. To make up for the lengthening of the critical path with time due to aging, Vdd has to continue to increase with time (top chart). The result is that the critical path takes the same time (bottom chart) and, therefore, the high frequency is maintained. However, since aging is exponential on Vdd and T (Equation 2.1), the high Vdd and resulting high T rapidly age the critical path. Soon, the aging is such that, to keep up with the high frequency required, Vdd (or T) would have to go above allowed values. At that point, shown as S_{SH} in Figure 2.3(c), the processor is discarded.

Finally, VSAM-Short is a simpler design than DVSAM-Short. Figure 2.3(d) shows its operation. Vdd is set to a value higher than Vdd_{NOM} (top chart of the figure). However, instead of dynamically compensating the increase in critical path delay due to aging with higher Vdd, we keep Vdd constant (top chart). As a result, the critical path delay increases (bottom chart). After a relatively short duration S_{SH} (different than for DVSAM-Short), the processor has aged too much and is discarded. Note that the critical path delay is not constant, while we want to keep the frequency constant. Consequently, the processor can only be clocked at the frequency allowed by the path delay at S_{SH}, namely τ_{OP} in the figure.

2.4 The BubbleWrap Many-Core

2.4.1 Overview

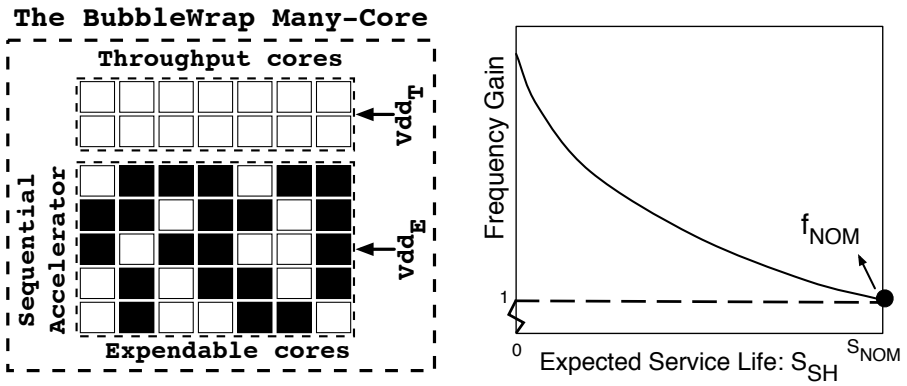
The *BubbleWrap* is a novel many-core that uses DVSAM to address the problem described in Section 2.1 of not being able to power-on all the on-chip cores simultaneously. While BubbleWrap uses all DVSAM modes, its most novel characteristic is the use of DVSAM-Short and VSAM-Short.

BubbleWrap has N architecturally homogeneous cores, of which only N_T can be powered-on simultaneously. The architecture distinguishes two groups of cores: *Throughput* (T) and *Expendable* (E) cores. In a die affected by process variation, we select N_T cores as the Throughput cores. We choose the ones that consume the least power at the target frequency. They are used to run the parallel sections of applications where, typically, we want throughput. The rest of the cores ($N_E = N - N_T$) are designated as Expendable cores. They are dedicated to run the sequential sections of applications, where we want per-thread performance.

The Expendable cores form a sequential accelerator. In our most novel designs, we attain acceleration by sacrificing one Expendable core at a time. Specifically, each Expendable core runs under DVSAM-Short (or VSAM-Short) mode, at elevated Vdd and frequency. It delivers high performance, but it ages fast until it can no longer sustain such conditions. At that point, we say that the core *pops*. It is discarded and replaced by another core from the Expendable group. This process of *popping* cores by applying DVSAM-Short (or VSAM-Short) modes gives its name to the BubbleWrap many-core.

Figure 2.4(a) shows a logical view of the BubbleWrap many-core. The figure depicts a mid-life

chip, when some of the Expendable cores have already popped (in black). Although the Throughput and Expendable cores form two logical groups, process variation determines their actual location on the die and, therefore, the cores of each group are typically not physically contiguous. All Throughput cores, however, receive the same supply voltage Vdd_T. When active, an Expendable core receives Vdd_E.



(a) (b)

Figure 2.4: BubbleWrap chip (a) and operation (b).

To estimate how quickly BubbleWrap can afford to pop Expendable cores, we add up the sequential-execution time of all the applications that are expected to run on the chip over its service life. This number, as a fraction of the nominal service life S_{NOM} of the chip, is called the *Sequential Load* (L_{SEQ}). For example, if we expect to run 21 applications, each of which runs a sequential section for 6 months, and the nominal service life of the chip is 7 years, then $L_{SEQ} = 21 \times 0.5/7 = 1.5$. Knowing L_{SEQ} and the number of Expendable cores N_E, we can conservatively estimate the short service life of each individual Expendable core S_{SH} as follows

$$S_{SH} = \frac{S_{NOM} \times L_{SEQ}}{N_E} \tag{2.7}$$

In our example, if we have 32 Expendable cores, each has to last $S_{SH} = 7 \times 1.5/32$, which is approximately 4 months. In reality, each core will take less than 4 months to execute its load because, thanks to its higher Vdd, it runs faster.

Figure 2.4(b) qualitatively shows how an Expendable core's shorter service life (S_{SH}) permits operation at increasingly higher frequencies. The figure plots the frequency gain over the nominal frequency as a function of S_{SH}. The curve is generated with representative parameter values. It

can be shown that the frequency gain increases exponentially with decreasing S_{SH}. Consequently, for (D)VSAM-Short to be profitable, it is required that $S_{SH} \ll S_{NOM}$. Fortunately, we expect that S_{SH} will continue to shrink with time, since technology scaling is providing more Expendable cores with each generation.

2.4.2 BubbleWrap Environments

The application of the different DVSAM modes of Section 2.3 to the Throughput or Expendable cores gives rise to six different BubbleWrap environments. The environments are pictorially shown in Figure 2.5 and described in Table 2.2.

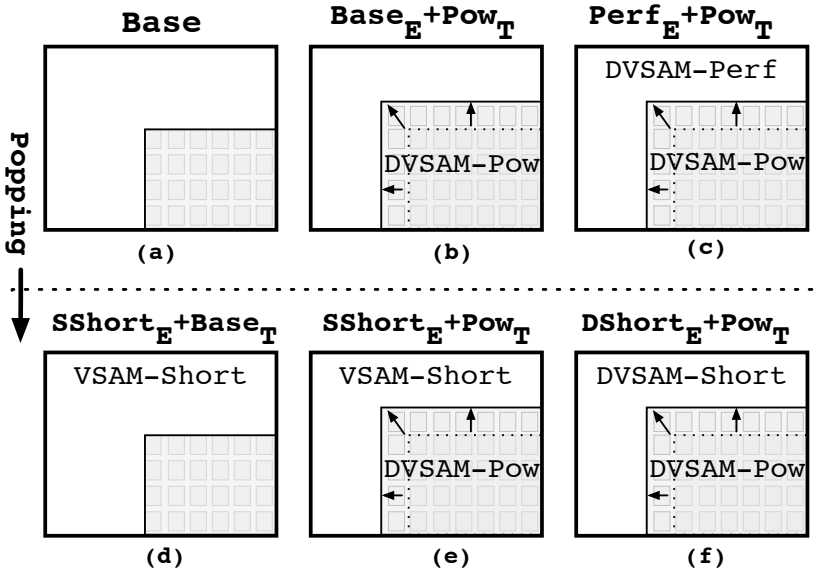


Figure 2.5: BubbleWrap chips corresponding to different environments. In each environment, Throughput cores are in gray and Expendable cores in white. Recall that *popping* means applying DVSAM-Short or VSAM-Short to the Expendable cores.

Chip (a) in Figure 2.5 shows the *Base* environment, which serves as the baseline. In *Base*, we disable the Expendable cores and operate the Throughput cores at Vdd_{NOM} and f_{NOM}.

Chip (b) shows the $Base_E$+Pow_T environment, where we keep the Expendable cores disabled and apply DVSAM-Pow to the Throughput cores. Throughput cores operate at f_{NOM} and at a supply voltage less than Vdd_{NOM}. Since each Throughput core now consumes less power, we can *expand* the number of Throughput cores and power-on all of them at the same time. This is shown in Figure 2.5(b) with the arrows. Overall, this environment increases throughput while clocking

Environment	Vdd_E	Vdd_T
Base	N/A (Cores disabled)	Vdd_{NOM}
$Base_E$+Pow_T	N/A (Cores disabled)	$< Vdd_{NOM}$ (DVSAM-Pow)
$Perf_E$+Pow_T	Variable (DVSAM-Perf)	$< Vdd_{NOM}$ (DVSAM-Pow)
$SShort_E$+$Base_T$	$> Vdd_{NOM}$ (VSAM-Short)	Vdd_{NOM}
$SShort_E$+Pow_T	$> Vdd_{NOM}$ (VSAM-Short)	$< Vdd_{NOM}$ (DVSAM-Pow)
$DShort_E$+Pow_T	$> Vdd_{NOM}$ (DVSAM-Short)	$< Vdd_{NOM}$ (DVSAM-Pow)

Table 2.2: Voltage applied to the Expendable cores (Vdd_E) and to the Throughput cores (Vdd_T) in each of the BubbleWrap environments.

all the processors at f_{NOM}.

Chip (c) shows the $Perf_E$+Pow_T environment, where we apply DVSAM-Perf to the Expendable cores (one at a time, during sequential sections) and, as before, DVSAM-Pow to the Throughput cores (to all of them, under expansion, during parallel sections). Expendable cores now deliver higher sequential performance, while Throughput cores deliver higher throughput.

The environments in chips (d), (e), and (f) are the same as the ones in (a), (b), and (c) except that, in addition, we apply core popping. Recall that this means applying VSAM-Short or DVSAM-Short to the Expendable cores. As a result, these environments further increase sequential performance.

The type of popping we perform (VSAM-Short or DVSAM-Short) in each case depends on whether the corresponding chip in Figure 2.5 already supported DVSAM on Expendable cores to start with. Specifically, if it did not, we apply VSAM-Short; if it did, we apply DVSAM-Short. Consequently, in chip (d), we take *Base* and apply VSAM-Short to the Expendable cores. The result is $SShort_E$+$Base_T$. In chip (e), we take $Base_E$+Pow_T and apply VSAM-Short to the Expendable cores. The result is called $SShort_E$+Pow_T. Finally, in chip (f), we take $Perf_E$+Pow_T and apply DVSAM-Short to the Expendable cores, creating the $DShort_E$+Pow_T environment.

2.4.3 Hardware Support for BubbleWrap

We describe three hardware components of BubbleWrap, namely the power and clock distribution system, the aging sensors, and the optimizing controller.

Power and Clock Distribution System

Our proposed BubbleWrap design has two voltage and frequency domains in the chip, namely one for the set of Throughput cores and one for the set of Expendable ones. This simple implementation provides enough functionality for our environments of Figure 2.5. Indeed, in such environments, all the cores in the Throughput set operate under the same conditions, and these conditions are potentially different than those for the Expendable cores. Note that other designs are also possible. For example, in a scenario with high within-die process variation, it may make sense to have one voltage and frequency domain per core. Alternatively, in a scenario where the chip runs a single application at a time, which alternates between parallel and sequential phases, BubbleWrap would only need a single voltage and frequency domain. This simpler design would suffice because we would not have Throughput cores and Expendable cores busy at the same time. However, we do not consider such designs here.

Let us consider the power distribution first. Since the physical location of the Throughput and Expendable cores on chip is unknown until manufacturing test time, we need a design that can supply either Vdd_E or Vdd_T to any core on the die. The most flexible solution is to include two independent supply networks [26], and connect all cores to both grids through power-gating transistors. In this manner, a controller can select, for each core, whether to connect to the Vdd_E grid or the Vdd_T grid by turning on the appropriate transistor.

Figure 2.6(a) shows such a design. The figure shows a chip with a global controller and one core. The core has a multiplexer that can select one of the two power grids. The controller determines the values of Vdd_E and Vdd_T, along with which grid each core should be connected to. This design has the advantage of allowing cores to move from one grid to the other dynamically, possibly based on their aging conditions.

Figure 2.6(b) shows the clock distribution network. The figure shows the grid with a global controller and four cores — one Expendable and three Throughput ones. Each core contains a PLL for signal integrity. The controller manages each PLL so that the correct frequency is supplied. This design allows a core to move from one of the core sets to the other dynamically. Moreover, if needed, it can also support per-core frequency domains [5, 27].

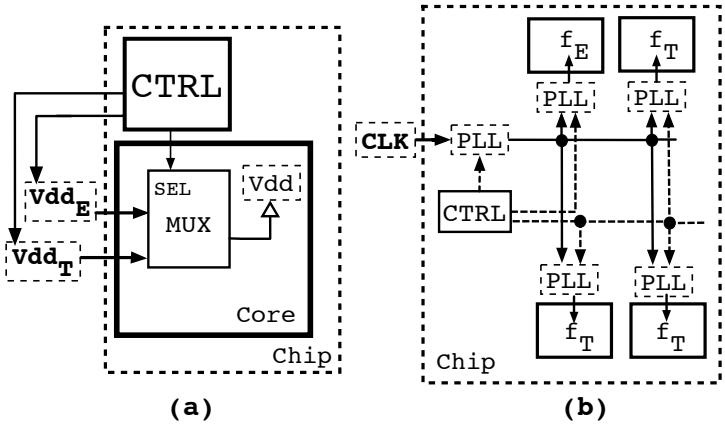


Figure 2.6: BubbleWrap power (a) and clock (b) distribution.

Aging Sensors

The effective application of the different DVSAM modes requires that we have a way to reliably estimate the aging that each core experiences. This can be done by using aging sensors that dynamically measure the increase in critical path delays due to aging. The literature proposes a variety of techniques for this purpose, including canary paths distributed throughout the cores or periodic BIST (e.g., [8,9,11,22–25]). It is claimed that such techniques can measure critical path delays with sub-μs measurement times and sub-ps precision [23]. Moreover, Intel Core i7 [5] already includes some of this circuitry to determine the level of Turbo Boost that can be applied.

Optimizing Dynamic Controller

The BubbleWrap controller interacts with the rest of the chip as shown in Figure 2.7(a). The controller performs several tasks. First, it dynamically adjusts the $Vdd_T(t)$ and $Vdd_E(t)$ supplied to the two sets of cores. Second, it keeps a table of which cores are Throughput, which are Expendable, and which ones are currently running. Based on this information, it sends the core selection signals described in Section 2.4.3. Note that, if it uses any of the BubbleWrap environments with core popping, the controller also determines when an Expendable core can be considered popped and keeps track of which cores are already popped. To perform all these tasks, the controller needs age information from all the cores.

We envision a simple hardware-based implementation of the controller. Every time that the operating system (OS) changes the threads running on the chip, it passes information to the con-

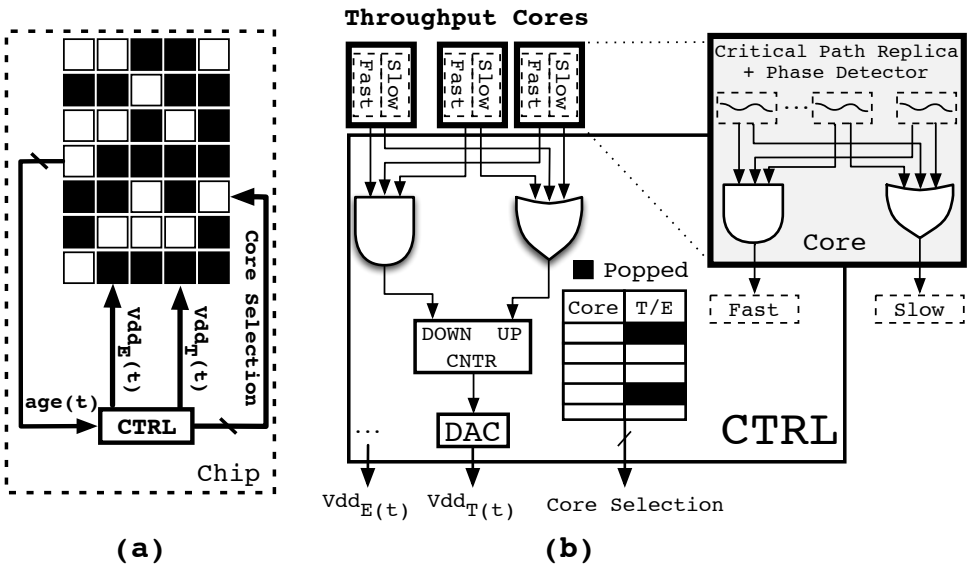


Figure 2.7: Overview of the BubbleWrap controller.

troller on which DVSAM mode is most appropriate for each thread, or at least which threads require acceleration. Based on this information and core age information, the controller outputs its initial Vdd_T, Vdd_E, and the core selection signals.

As the threads execute, the controller dynamically tunes the Vdd_T and Vdd_E values. At any time, its goal is to supply the *minimum* Vdd_T (or Vdd_E) value that enables the cores to keep up with the target operational frequency (f_{OP}) for the current DVSAM mode. Note that such frequency is not set by the controller; it is set statically by the manufacturer.

To tune the voltages, the controller relies on the age information provided by the cores in Figure 2.7(a). To see how this works, Figure 2.7(b) shows the internals of the controller. The figure also shows 3 Throughput cores, one of which is expanded.

In each core, aging sensors detect aging-induced increases in critical path delays. The figure sketches a design similar to the one by Teodorescu *et al* [**?**]. It uses multiple critical path replicas distributed across the core along with a phase detector. If each replica satisfies the frequency specification by a certain margin, the core asserts signal *Fast*, to indicate that this core may operate at a lower Vdd and still be clocked at the target frequency. If at least one of the replicas does not satisfy the frequency specification, signal *Slow* is asserted to demand a higher Vdd for this core to support the target frequency. If no signal is set, all of the replicas satisfy the frequency specification with the lowest margin. The combination of the Fast and Slow signals is the age

information passed from each core to the controller.

The controller collects all Fast and Slow signals from all Throughput cores and combines them using a similar circuit inside the controller (Figure 2.7(b)). The circuit asserts signal *DOWN* when a lower Vdd may be feasible to cycle at the target frequency; it asserts signal *UP* if at least one of the cores requires a higher Vdd. If no signal is asserted, all of the cores satisfy the frequency at the lowest possible power budget. Finally, *DOWN* and *UP* form the control inputs to a counter. The counter is then connected to a digital-to-analog converter (DAC), which converts the counter output to Vdd_T.

Although not shown in the figure, an analogous circuit exists for Expendable cores. In addition, the figure shows the table that records the mapping of cores to Throughput or Expendable type, and which cores are already popped.

The characteristics of the DAC, together with the magnitude of the voltage noise determine the grain size at which the DVSAM modes can tune Vdd.

2.5 Evaluation Setup

In this section, we present the evaluation methodology that we use to characterize a BubbleWrap chip in a near-future process technology.

Cox	$2.5 \times 10^{-20} F/nm^2$	n	1.5
k_T	$-1mV/K$	E_0	$0.08V/nm$
T_0	70^oC	Ea	$0.56eV$
T_{MAX}	100^oC	a	0.2
Vdd	$0.8 - 1.3V$	S_{NOM}	7 years
Vdd_{NOM}	$1V$	α	1.3
Vth_{NOM}	$250mV$	G	10% at T_{MAX}
k_{DIBL}	$-150mV/V$	η	0.35

Table 2.3: Technology parameters.

2.5.1 Power and Thermal Model

We use a simple model to estimate power and temperature values in a BubbleWrap chip. We estimate the dynamic power consumed by a core using Wattch [28]; for the static power, we use

Technology node: 22nm Cores per chip: 16 T + 16 E	f_{NOM}: 4.5GHz P_{MAX}: 80W (all cores), 5W (T core), 10W (E core)
Width: 6-fetch 4-issue 4-retire OoO ROB: 152 entries Issue window: 40 fp, 80 int LSQ Size: 54 LD , 46 ST Branch pred: 80Kb tournament	L1 D Cache: 16KB WT, 0.44ns round trip, 4 way, 64B line L1 I Cache: 16KB, 0.44ns round trip, 2 way, 64B line L2 Cache: 2MB WB per core, 2ns round trip, 8-way, 64B line, has stride prefetcher Memory: 80ns round trip

Table 2.4: Microarchitectural parameters.

the following equations:

$$\begin{aligned} P_{STA} &= Vdd \times I_{LEAK} \\ I_{LEAK} &\propto \mu \times T^2 \times e^{-qVth/kTn} \quad , where \;\; \mu \propto T^{-1.5} \end{aligned} \tag{2.8}$$

Adding up the power consumed by all of the on-chip cores in P_{TOT}, we use the following equations to estimate the (junction) temperature T_J of a core:

$$\begin{aligned} T_J &= T_S + \theta_{JS} \times (P_{STA} + P_{DYN}) \quad \textit{(per core)} \\ T_S &= T_A + \theta_{SA} \times P_{TOT} \end{aligned} \tag{2.9}$$

In the equations, θ_{SA} is the spreader to ambient thermal resistance, which is modeled as a lumped equivalent of spreader to heat-sink and heat-sink to ambient thermal resistances. Moreover, T_A is the ambient temperature, T_S is the spreader temperature, P_{STA} and P_{DYN} are the static and dynamic power consumptions of the core, and θ_{JS} is the junction to spreader thermal resistance. We assume an ambient temperature T_A of 45°C and a θ_{SA} of 0.222K/W [5], while θ_{JS} is calibrated for the worst case operating conditions.

For the near-future technology node we are assuming, the core area is already so small that intra-core spatial thermal variation becomes negligible. Therefore, we model temperature at core granularity, which offers reasonable fidelity for many-core designs [29]. Moreover, since we assume a checkerboard core-cache layout, which reduces core-to-core thermal coupling significantly, we neglect inter-core lateral conduction. The cache power density is significantly lower than the core power density. Hence, caches placed between cores act as virtual lateral heat-sinks.

We impose a maximum chip power in the cores of 80W, and a maximum power of 5W per Throughput core. A core is designed to support a power limit of 10W; going above that could damage its power-distribution system. Consequently, 10W is the effective maximum power for

an Expandable core.

2.5.2 Process Technology

We assume a 22nm technology node based on the Predictive Technology Model's bulk HK-MG process [30], which incorporates recent corrections [31]. The technology parameters are given in Table 2.3.

We use a very simple model to estimate the effects of Vth process variation. Power and performance asymmetry due to process variation at the core granularity stems mainly from systematic variation, which shows strong spatial dependence [32]. Hence, given the small area taken by a core, we neglect any intra-core Vth variation. Moreover, we assume a normally-distributed core-to-core variation in Vth with $\sigma/\mu = 5\%$. In our evaluation, we perform some experiments on cores with $[-3\sigma, +3\sigma]$ deviation from Vth_{NOM}. In our experiments, such range causes a variation of approximately 12% in power consumption between the most power consuming core and the least consuming one. Moreover, it leads to a variation of approximately 14% in frequency between the fastest core and the slowest one.

In our experiments, we assume that processors have a nominal service life S_{NOM} of 7 years. The constant of proportionality of Equation 2.1, A_{BTI}, is calibrated so that a core with a $+3\sigma$ Vth value (i.e., the slow corner) slows down by $G = 10\%$ at the end of S_{NOM} if operated continuously at Vdd_{NOM} and T_{MAX}. The constant of proportionality for the alpha-power law (Equation 2.3) is calibrated to guarantee operation at f_{ZG} for a core with a $+3\sigma$ Vth value at the beginning of the service life, at Vdd_{NOM} and T_0. Finally, the constant of proportionality for leakage current (Equation 2.8) is set so that leakage accounts for 25% of the total power consumption for a core with a -3σ Vth value (i.e., the leaky corner) at Vdd_{NOM} and T_0.

2.5.3 Workload

Each non-idle core repeatedly cycles through the SPECint2000 applications in a round-robin fashion (switching every $\approx$ 45minutes of simulated time), therefore experiencing a diverse range of work over its service life. We create parallel sections, where all Thoughput cores are busy running this load (or an expanded number of Thoughput cores). We also create sequential sections, where only one Expendable core is running. To assess how BubbleWrap performs with different propor-

tions of parallel and sequential sections, we parameterize the workload with the fraction W_{SEQ} of time spent in the sequential section — for the default number of Throughput cores, without expansion. The evaluation characterizes BubbleWrap for $W_{SEQ} = [0, 1]$.

2.5.4 Many-Core Microarchitecture

We model a near-future 32-core many-core microarchitecture as described in Table 2.4. Based on ITRS projections [4] (along with corrections based on current industry status), we use 16 Throughput and 16 Expendable cores as the default (not counting expansion for the set of Throughput cores). We simulate the core microarchitecture with the SESC simulator [33] instrumented with Wattch models [28].

2.6 Evaluation

In this section, we assess the impact of BubbleWrap in terms of power and performance. The evaluation optimistically assumes that the aging sensors always give correct information on the critical path length and that the BubbleWrap controller adds no overhead. In the following, we analyze DVSAM-Pow, DVSAM-Perf, DVSAM-Short and VSAM-Short, and finally the different BubbleWrap environments.

2.6.1 Enhancing Throughput: DVSAM-Pow

For the evaluation, we consider three cores to cover the whole Vth spectrum: one with Vth_{NOM}, one with Vth at -3σ, and one with Vth at $+3\sigma$. They are all clocked at f_{NOM} and work at workload temperature conditions. To each of these cores, we apply DVSAM-Pow. The second column of Table 2.5 shows, for each of the cores, the energy reductions attained with DVSAM-Pow over the entire service life of the core. From the table, we see that DVSAM-Pow reduces the energy consumption by 13–31%. The savings are more pronounced for the cores with low Vth, which suffer from higher static energy consumption. For the core with Vth_{NOM}, the savings are a substantial 23%.

We now take the core with Vth_{NOM} before and after the optimization and plot its temporal evolution over the whole service life. We call the two resulting cores *Base* and *DVSAM-Pow*, respectively, and show the Vdd evolution (Figure 2.8(a)), normalized critical path delay τ/τ_{ZG} evolution

Deviation in Vth	Energy Savings due to DVSAM-Pow (%)	Frequency Increases due to DVSAM-Perf (%)
-3σ	31	18
0	23	14
$+3\sigma$	13	10

Table 2.5: Benefits of DVSAM-Pow and DVSAM-Perf.

(Figure 2.8(b)), power evolution (Figure 2.8(c)), and temperature evolution (Figure 2.8(d)). The plots show banded structures for the curves. They are due to temporal variations in the workload, as we execute the SPECint2000 applications in a round-robin manner.

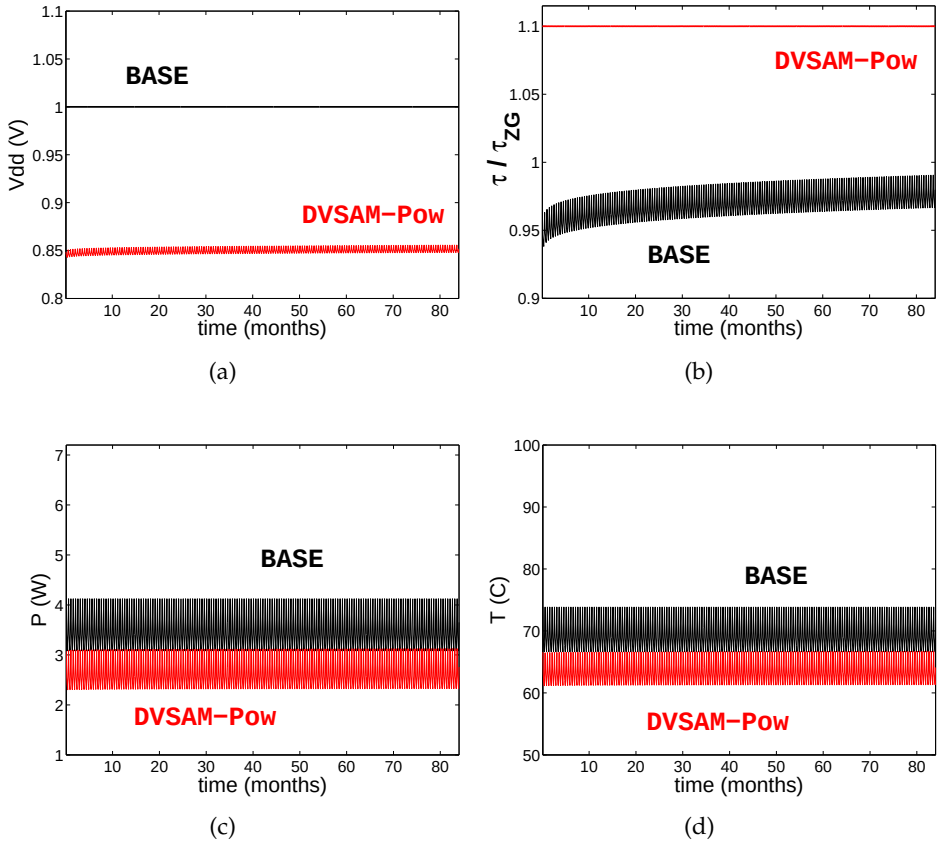


Figure 2.8: Temporal evolution of the effects of DVSAM-Pow.

Figure 2.8(a) corresponds to the top row of Figure 2.3(a). We see that DVSAM-Pow keeps Vdd about 0.15V lower than Base. Moreover, the difference does not decrease much over the whole lifetime. To understand why, consider Figure 2.8(b), which corresponds to the second row of

Figure 2.3(a). While DVSAM-Pow's critical path consumes all the guard-band from the beginning (by design), Base's critical path delay increases only a little, never consuming much of the guard-band. The reason is because the guard-band is dimensioned for the worst case conditions, namely a core with Vth at $+3\sigma$ operating at T_{MAX}. In our case, Base has Vth_{NOM} and operates at workload temperatures. As a result, it does not age as much and does not use most of the guard-band. Overall, the across-the-board gap in Figure 2.8(b) induces the resulting gap in Figure 2.8(a).

Note that DVSAM-Pow only consumes the guard-band set aside for aging (and variation). There is additional guard-banding present for other reasons, such as voltage noise. That one remains.

Figure 2.8(c) shows that the lower operating voltage of DVSAM-Pow saves significant power compared to the core operating continuously at Vdd_{NOM}. Finally, Figure 2.8(d) shows that the temperature also decreases due to the lower operating voltage.

2.6.2 Enhancing Frequency: DVSAM-Perf

We now take the three cores covering the Vth spectrum as explained in the beginning of Section 2.6.1 and apply DVSAM-Perf. The third column of Table 2.5 shows, for each of the cores, the frequency increases attained with DVSAM-Perf over f_{NOM}. From the table, we see that DVSAM-Perf increases the frequency by 10–18%. The increase is larger for the core with low Vth, since this core can cycle faster for any given Vdd. For the core with Vth_{NOM}, the increase is a significant 14%.

Using the core with Vth_{NOM} before and after the optimization, we plot its temporal evolution over the whole service life. We call the resulting two cores *Base* and *DVSAM-Perf*, respectively, and show the evolution of Vdd (Figure 2.9(a)), normalized critical path delay τ/τ_{ZG} (Figure 2.9(b)), power (Figure 2.9(c)), and temperature (Figure 2.9(d)).

Figure 2.9(a) corresponds to the top row of Figure 2.3(b). We see that, with DVSAM-Perf, Vdd starts around Vdd_{NOM} at the beginning of the service life and increases beyond Vdd_{NOM} thereafter. At the end of the service life, it reaches around 1.25V. Consider now Figure 2.9(b), which corresponds to the second row of Figure 2.3(b). Thanks to DVSAM-Perf's high Vdd operation, DVSAM-Perf keeps the critical path delay around $0.96\times\tau_{ZG}$. As a result, DVSAM-Perf operates at a constant frequency that is 14% higher than f_{NOM} over the whole service life. Recall that f_{NOM} is the frequency of the Base core. It has a period of $\tau_{ZG} \times (1 + G) = \tau_{ZG} \times 1.1$. This substan-

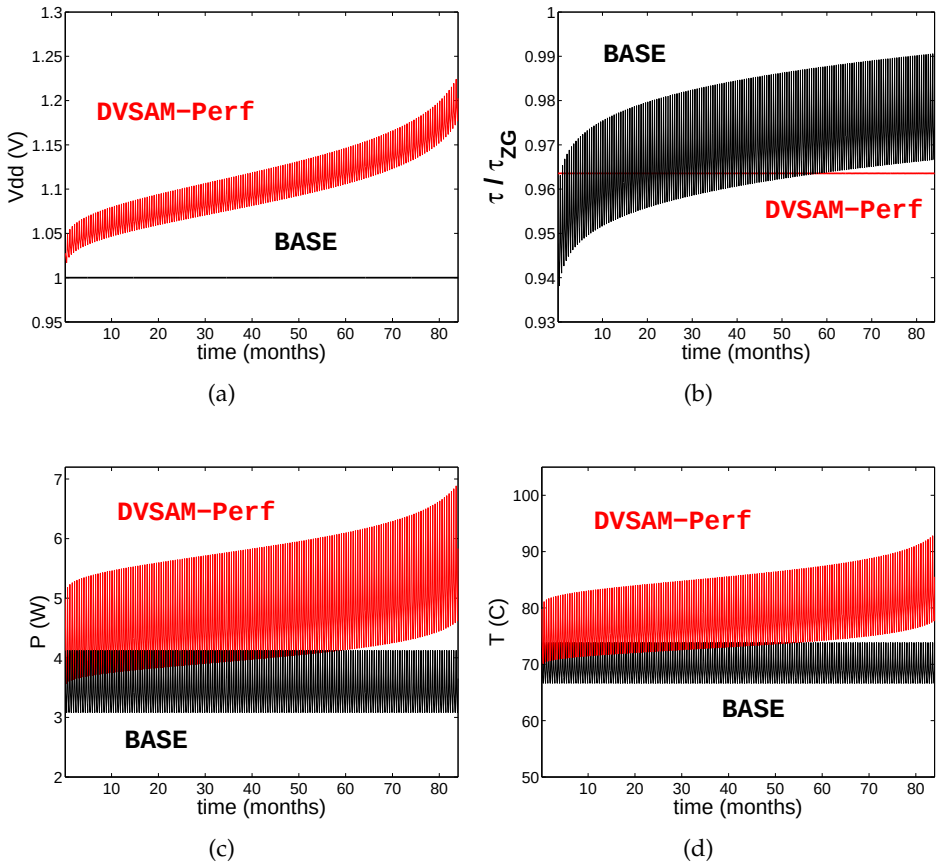


Figure 2.9: Temporal evolution of the effects of DVSAM-Perf.

tial frequency increase, even over the frequency of no guard-band ($1/\tau_{ZG}$) is possible because the guard-band is dimensioned for a core with Vth at $+3\sigma$ operating at T_{MAX}.

Figure 2.9(c) shows that the higher voltages of DVSAM-Perf cause a continuous increase in core power. By the end of the service life, the power consumed by a core is 7W. This is a high value, but still less than the 10W reserved to Expendable cores. As shown in Figure 2.9(d), the junction temperature goes above 90°C at the end of the service life.

2.6.3 Popping: DVSAM-Short & VSAM-Short

We now consider the effect of popping Expendable cores, first with DVSAM-Short and then with VSAM-Short. As before, we use a core with Vth_{NOM}. We are interested in the evolution of the core at elevated voltage and frequency conditions for a period S_{SH} that ranges from 0 to S_{NOM}.

We consider DVSAM-Short first. Figure 2.10 considers all possible values of S_{SH} and shows: The maximum Vdd that gets applied to the core (Chart (a)), the frequency of the core relative to f_{NOM} (Chart (b)), the maximum power consumed by the core (Chart (c)), and the maximum temperature attained by the core (Chart (d)). Each chart highlights two data points, namely one for a one-month service life (labeled *1mo*) and one for a nominal service life (labeled S_{NOM}). The latter corresponds to the DVSAM-Perf mode.

Recall from the top row of Figure 2.3(c) that, under DVSAM-Short, Vdd starts-off elevated and continues to increase until the core pops (i.e., we need to violate Vdd_{MAX}, P_{MAX}, or T_{MAX} to maintain the frequency). Figure 2.10(a) shows that, in all cases of S_{SH}, Vdd practically reaches Vdd_{MAX}, which is 1.3V. Figures 2.10(c) and 2.10(d) show that the maximum core power and temperature, respectively, are fairly similar for different S_{SH}, and have not reached their allowed limit — core power is 6.5–7W and temperature is 90–95°C.

Finally, Figure 2.10(b) shows that the frequency at which we can clock the core increases as S_{SH} becomes smaller. For a service life of 1 month, the core can be clocked at a frequency of 1.19 relative to f_{NOM}. However, as S_{SH} increases, the frequency quickly goes down. Very soon, we attain frequencies not much higher than the one reached by DVSAM-Perf, namely 1.14 relative to f_{NOM}. This is because the Vdd conditions required to deliver high frequency quickly age the core, limiting its service life.

We now consider VSAM-Short. This mode represents a simpler approach than DVSAM-Short because we do not need to repeatedly adjust Vdd based on measurements of the critical path

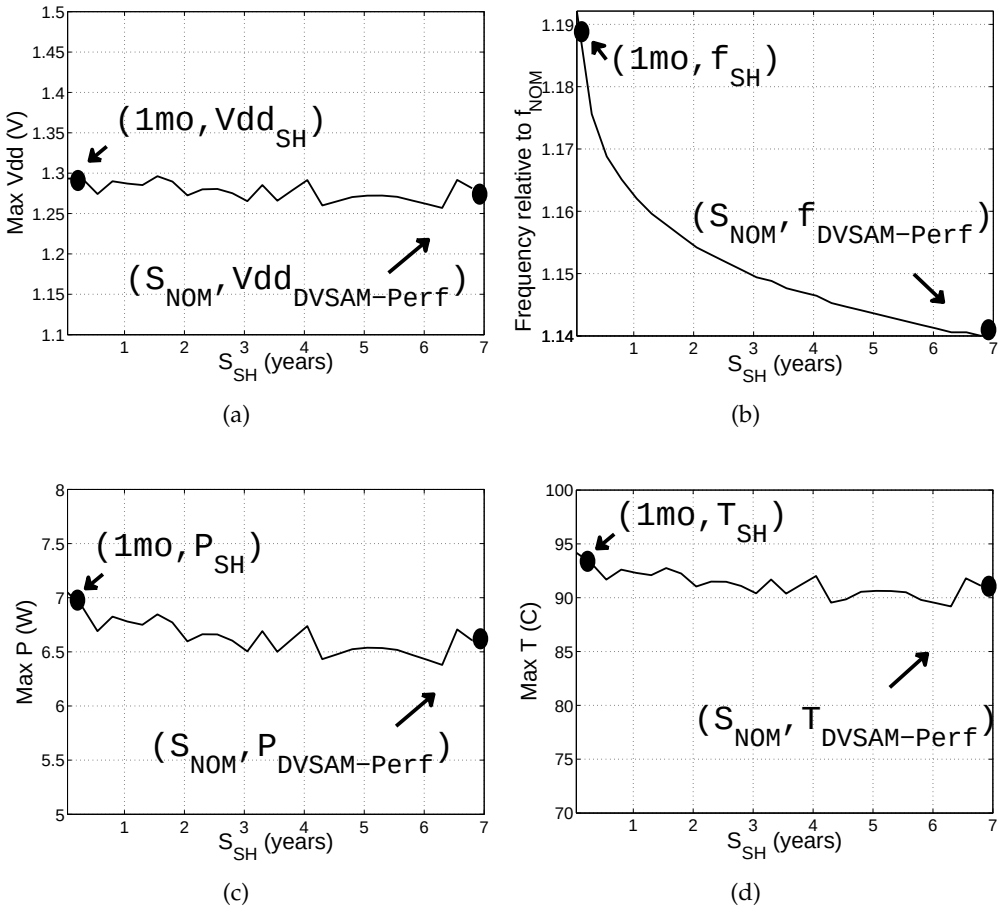


Figure 2.10: Impact of core popping with DVSAM-Short.

delays. We simply set a constant, elevated Vdd_{SH} for the duration of the shorter service life S_{SH}. This was shown in the top row of Figure 2.3(d). However, this mode is less effective than DVSAM-Short because we need to set Vdd_{SH} conservatively — with the same conservative assumptions as we set Vdd_{NOM} when we want the processor to last for S_{NOM}. Specifically, Vdd_{SH} should guarantee that, if operated continuously at T_{MAX}, the core with $+3\sigma$ Vth deviation (the slow corner) consumes the entire guard-band only at the end of its presumed short service life (S_{SH}).

With these assumptions, we generate Figure 2.11(a), which shows for each short service life S_{SH}, the elevated Vdd_{SH} that we need to apply. The figure highlights two data points, namely one for a one-month service life (labeled *1mo*) and one for the nominal service life (labeled S_{NOM}). The latter corresponds to a core operating at Vdd_{NOM}. We can see that, for $S_{SH} = 1$ month, we get a Vdd_{SH} of only 1.1V.

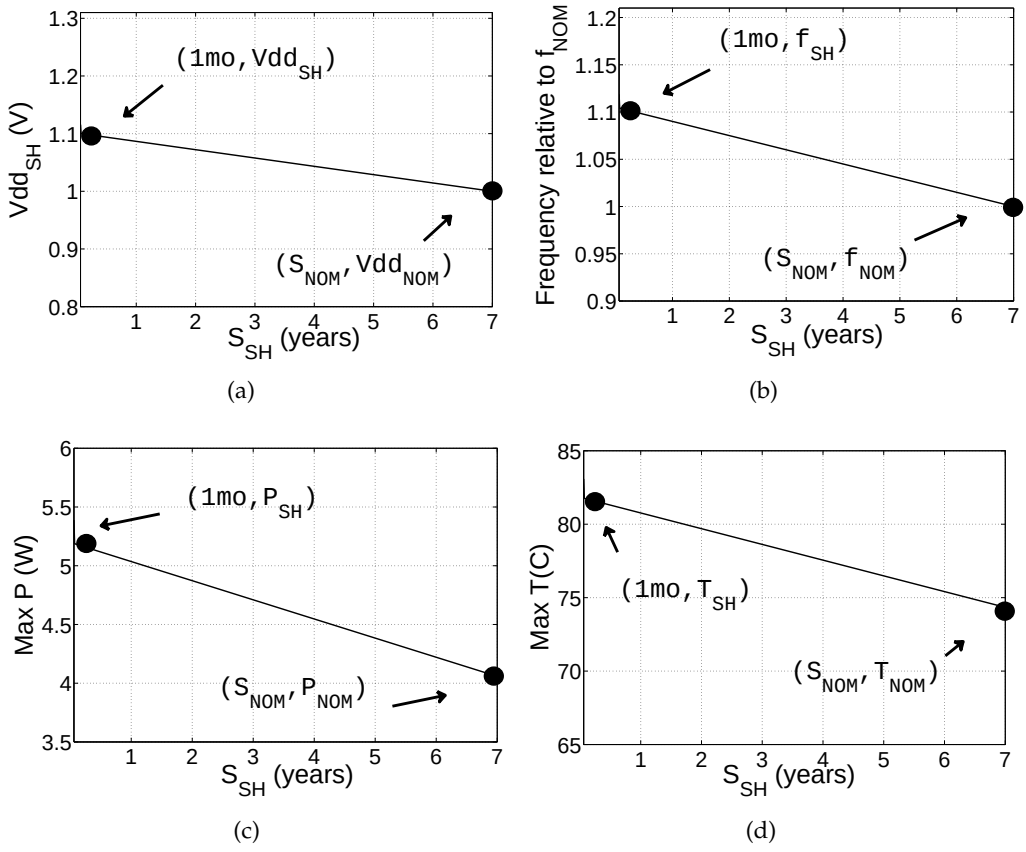


Figure 2.11: Impact of core popping with VSAM-Short.

Figures 2.11(b), (c), and (d) repeat Figures 2.10(b), (c), and (d) for VSAM-Short. As usual, we characterize a core with Vth_{NOM}. From Figure 2.11(b), we see that the frequency at which we

can clock the core increases as S_{SH} becomes smaller. However, VSAM-Short does not reach the frequency values that DVSAM-Short attains in Figure 2.10(b). At a service life of 1 month, VSAM-Short clocks the core at a frequency of 1.1 relative to f_{NOM}, in contrast to DVSAM-Short's 1.19.

Figures 2.11(c) and 2.11(d) show that the maximum power and temperature reached by an Expendable core with a short service life of 1 month are around 5.2W and around 82°C, respectively. This is in contrast to the higher values attained with DVSAM-Short, namely 7W and 93°C (Figures 2.10(c) and 2.10(d)).

2.6.4 BubbleWrap Environments

We now estimate the performance and power impact of each of the BubbleWrap environments described in Figure 2.5 and Table 2.2. BubbleWrap increases the frequency of sequential sections by popping Expendable cores, and the throughput of parallel sections by expanding the number of Throughput cores. Next, we examine the frequency of sequential sections, the throughput of parallel sections, and finally the performance and power of the application as a whole.

Sequential Section Frequency

The frequency increase achievable by popping cores is a function of the service life per Expendable core (S_{SH}). From Equation 2.7, S_{SH} depends on the sequential load (L_{SEQ}). The smaller L_{SEQ} is, the shorter is S_{SH}, and the larger is the frequency boost provided by each Expendable core.

To study a range of S_{SH}, we take our workload from Section 2.5.3 and vary the fraction of its original execution time in the sequential section (W_{SEQ}), from 1 to 0. Note that, for our workload, $L_{SEQ} = W_{SEQ}$. Then, for different values of W_{SEQ}, Figure 2.12 shows the frequency attained by the sequential section relative to f_{NOM} in all the BubbleWrap environments of Figure 2.5.

There are three groups of environments. First, there are the environments that do not use Expendable cores (*Base* and $Base_E + Pow_T$); these cannot get any increase in frequency. The second group is the environment that uses Expendable cores but does not pop them ($Perf_E + Pow_T$). Expendable cores are expected to last for S_{NOM} and, for performance, are run in DVSAM-Perf mode. As a result, $Perf_E + Pow_T$ increases the frequency of the sequential section by a fixed amount irrespective of W_{SEQ}. We see that this increase is 14%, and was already shown in Table 2.5.

Finally, there are the environments that pop Expendable cores ($SShort_E + Base_T$, $SShort_E + Pow_T$ and $DShort_E + Pow_T$). These environments increase the sequential section frequency. The

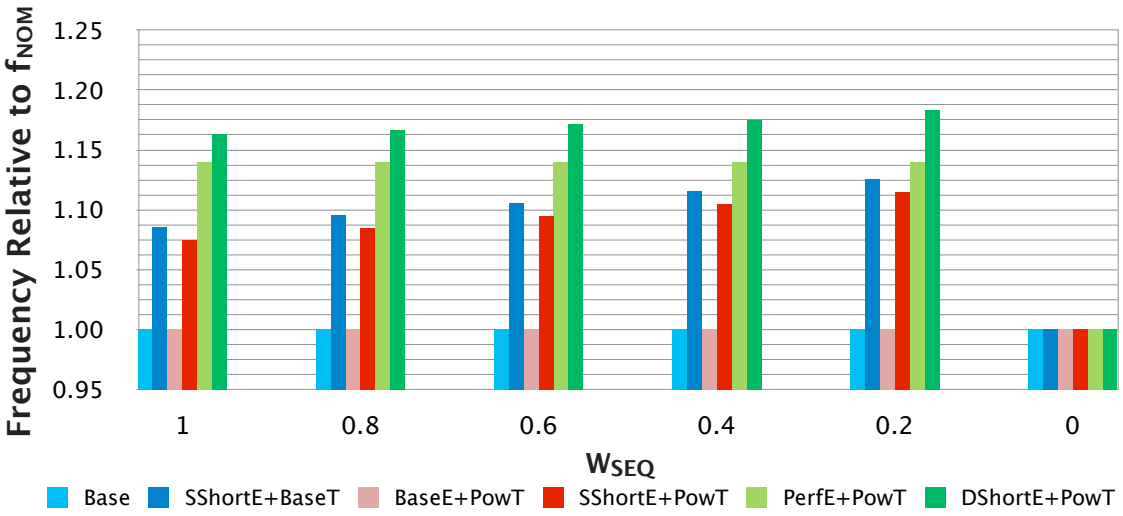


Figure 2.12: Frequency of the sequential section for each environment.

lower W_{SEQ} is, the higher their impact is, except for $W_{SEQ} = 0$, where they have no impact. Among the three environments, the one that uses DVSAM-Short (rather than VSAM-Short), is the one with the highest impact : $DShort_E + Pow_T$. For a fully sequential execution ($W_{SEQ} = 1$), we see that this environment provides a 16% frequency increase. Finally, $SShort_E + Pow_T$ increases the frequency less than $SShort_E + Base_T$ because, in $SShort_E + Pow_T$, some Expendable cores are turned into Throughput ones during the parallel section when the set of Throughput cores expands.

Parallel Section Throughput

The throughput increase achievable by BubbleWrap environments during parallel sections depends on whether or not they can expand the set of Throughput cores. There are two groups of environments. First, there are the environments that do not expand the set of Throughput cores (*Base* and $SShort_E + Base_T$); these cannot get any increase in throughput. The other group is the rest of environments ($Base_E + Pow_T$, $SShort_E + Pow_T$, $Perf_E + Pow_T$ and $DShort_E + Pow_T$); they all use the DVSAM-Pow mode on Throughput cores and, therefore, expand them equally during parallel sections. Assuming that the parallel section has enough parallelism, we can increase the number of Throughput cores to make up for the power saved by DVSAM-Pow. As shown in Figure 2.8(c), DVSAM-Pow reduces the power of a core from 4.1W to 3.1W on average. This is a reduction of 24%. Consequently, for constant power, the number of Throughput cores to execute

the parallel section can be increased from the original 16 cores to $16 \times 4.1/3.1 \approx 21$ cores. This represents a throughput increase of $s_t \approx 30\%$ for these BubbleWrap environments.

Estimated Overall Speedup and Power Cost

From the previous two sections, we can now roughly estimate the overall speedup and the power cost of each BubbleWrap environment. We consider a workload that has a fraction of sequential time W_{SEQ}. For each BubbleWrap environment, we assume that the time of the parallel section scales down perfectly with the corresponding throughput increase s_t of Section 2.6.4; similarly, we assume that the time of the sequential section scales down perfectly with the relative frequency increase f_r of Figure 2.12. With these optimistic assumptions, we obtain the following execution time speedup over *Base*

$$Speedup = \frac{Time_{unoptimized}}{Time_{optimized}} = \frac{1}{W_{SEQ}/f_r + (1 - W_{SEQ})/s_t} \tag{2.10}$$

Figure 2.13 shows these speedups for all the BubbleWrap environments and different W_{SEQ} values. The speedups are normalized to *Base*. There are three groups of environments. First, $SShort_E + Base_T$ only speeds-up sequential sections; its impact decreases with W_{SEQ}. Second, $Base_E + Pow_T$ only speeds-up parallel sections and, therefore, its effect goes down as W_{SEQ} increases. Finally, the remaining three environments speed-up both sequential and parallel sections. They are the environments that show the highest speedups across most of the W_{SEQ} range. Their relative speedups follow their relative increase in sequential section frequency in Figure 2.12. Overall, $DShort_E + Pow_T$ gives the highest speedups, which range from 1.16 to 1.30.

Note that the best performing environment ($DShort_E + Pow_T$, which supports core popping) is not any more complex than the next best-performing one ($Perf_E + Pow_T$, which does not support core popping). Both environments use two supply networks, tune Vdd_E and Vdd_T, and have aging sensors in both Expendable and Throughput cores. However, the third best-performing environment ($SShort_E + Pow_T$) is significantly simpler, since it does not require tuning Vdd_E or keeping aging sensors for Expendable cores.

Finally, Figure 2.14 shows the power consumption of the different BubbleWrap environments normalized to the power of *Base*. The power cost of BubbleWrap operation stems only from se-

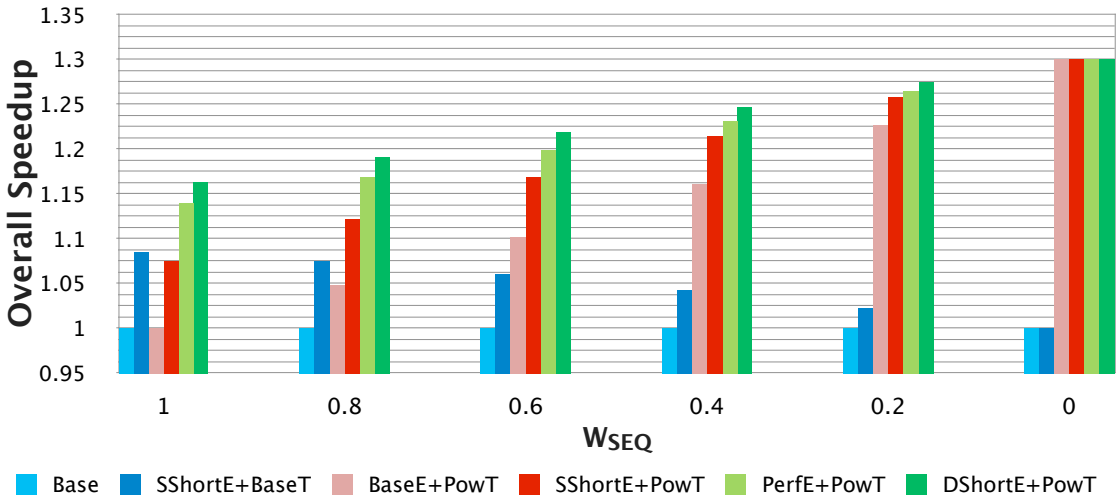


Figure 2.13: Speedup of BubbleWrap environments.

quential section acceleration — since the expansion in the number of Throughput cores has no power cost. Consequently, the normalized power increases with W_{SEQ}. Moreover, the figure shows that the most power-consuming environments are those that use the DVSAM-Short (or DVSAM-Perf) modes — namely, $DShort_E + Pow_T$ and $Perf_E + Pow_T$. This is because these modes raise Vdd_E to high values. On the other hand, the environments that use the VSAM-Short mode ($SShort_E + Base_T$ and $SShort_E + Pow_T$) consume much less power. This is because VSAM-Short does not tune Vdd_E and, therefore, has to set Vdd_E to conservatively low values.

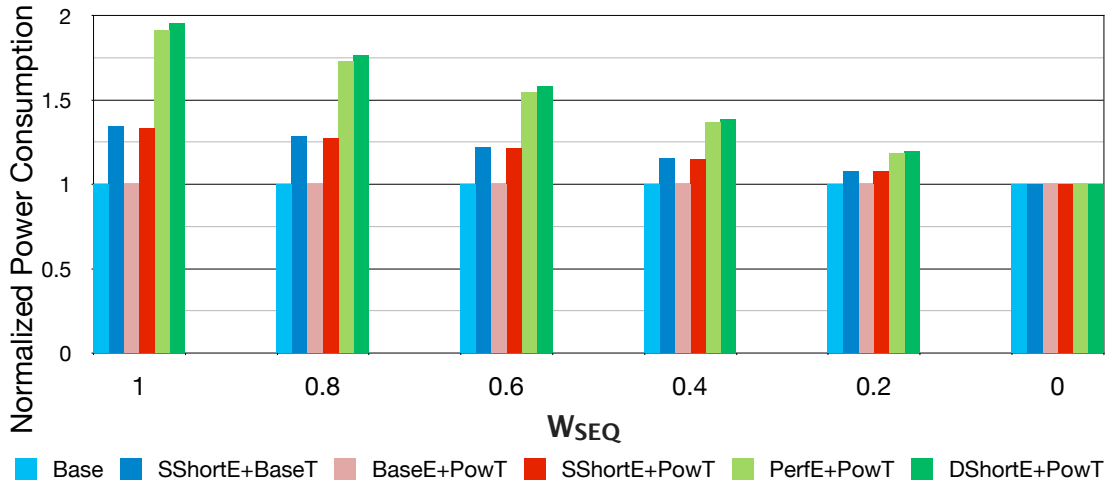


Figure 2.14: Power consumption of BubbleWrap environments.

2.6.5 Discussion

All the data shown in our evaluation except for Table 2.5 corresponds to processors with Vth_{NOM}. In reality, the cores in a BubbleWrap chip will exhibit process variation, which will affect how they respond to the BubbleWrap modes as shown in Table 2.5. This makes some of our results slightly optimistic while others slightly pessimistic. Specifically, the best cores will be chosen for the Throughput set. This is likely to result in higher parallel section speedup than reported here. However, it will also leave the worst cores for the Expendable set, which will result in lower sequential section speedup (or higher power consumption) than reported here. A further study should be done to address this issue.

We constrained the operation of all cores, including those that pop, to be within the T_{MAX} and Vdd_{MAX} envelopes. Respecting T_{MAX} and Vdd_{MAX} ensures a reliable service life of S_{NOM}. In reality, for a core that will have a short life, it may be possible to go above T_{MAX} or Vdd_{MAX}. This could allow core popping to deliver higher performance than reported here — perhaps at the cost of risking new types of hard or soft failures. This topic will be studied in future work.

2.7 Related Work

The impending many-core power wall is well-known in industry and reflected in recent ITRS projections [4]. However, our suggestion to expend (or pop) the excess cores that cannot be powered-on is novel as far as we know. Another proposal to address the many-core power wall is to use extremely low supply voltages, possibly close to the threshold voltage [34, 35]. This environment would allow all cores to operate simultaneously, albeit at severely reduced frequencies. Yet another alternative is a design comprising power-efficient, heterogeneous application-specific accelerators [36]. BubbleWrap is unique in extending the scaling of homogeneous many-cores without requiring core modifications.

Recently, processor aging has been the subject of interest in the computer architecture community (e.g., [7–14]). Some of this work attempts to reduce aging by setting the logic to recovery mode [7, 10]. However, the aging work most closely related to ours is Facelift [14]. Facelift proposed applying a few discrete voltage levels to minimize aging and showed how shorter service lifes could be exploited to increase core frequency. BubbleWrap's DVSAM framework improves on these techniques with continuous voltage tuning and with the power-saving DVSAM-Pow

mode. Additionally, BubbleWrap proposes a set of novel architectures (or environments) that use DVSAM. Finally, several authors have designed circuits that detect when a critical path has slowed down [8, 9, 11, 22–25]. Some of the BubbleWrap environments use these aging sensors.

2.8 Summary

To push back the many-core power wall, we made two main contributions. First, it introduced *Dynamic Voltage Scaling for Aging Management* (DVSAM) — a new scheme for managing processor aging by tuning Vdd (but not the frequency), exploiting any currently-left aging guard-band. The goal can be one of the following: consume the least power for the same performance and service life; attain the highest performance for the same service life and within power constraints; or attain even higher performance for a shorter service life and within power constraints.

Second, we presented *BubbleWrap*, a novel many-core architecture that makes extensive use of DVSAM to push back the many-core power wall. BubbleWrap selects the most power-efficient set of cores in the die — the largest set that can be simultaneously powered-on — and designates them as Throughput cores. They are dedicated to parallel-section execution. The rest of the cores are designated as Expendable. They are dedicated to accelerating sequential sections. BubbleWrap applies DVSAM in several environments. In some of them, it attains maximum sequential acceleration by sacrificing Expendable cores one at a time, running them at elevated Vdd for a significantly shorter service life each, until they completely wear-out.

In simulated 32-core chips, BubbleWrap provides substantial gains over a plain chip with the same power envelope. On average, our most aggressive design runs fully-sequential applications at a 16% higher frequency, and fully-parallel ones with a 30% higher throughput. We are now extending DVSAM to also include changes in processor frequency with time. This improvement should deliver better design points. We are also working on a model to accurately capture the deviations from the nominal behavior within a logical set of cores.

3 A MICROARCHITECTURAL MODEL OF PROCESS VARIATIONS FOR NTC

3.1 Introduction

Power or energy consumption is typically the primary concern in today's computer platforms, ranging from datacenters to handhelds. The main reason for their importance is that CMOS technology has long ago stopped scaling close to perfectly and, as a result, power density increases significantly with each technology generation. If we are to continue delivering scalable computing performance, we need to find new ways to compute more energy- and power-efficiently.

One way to attain higher energy efficiency is to reduce the supply voltage (Vdd) to a value only slightly higher than a transistor's threshold voltage (Vth). This environment is called Near-Threshold Computing (NTC) [37–39] — as opposed to conventional Super-Threshold Computing (STC). Vdd is a most powerful knob because it impacts both dynamic and static energy super-linearly. Current indications suggest that NTC can decrease the energy per operation by several times over STC [37, 38]. A drawback is that it imposes a frequency reduction, which may be tolerable through more parallelism in the application. For parallel loads, since more cores can be running concurrently within the chip's power envelope, the result is a higher throughput.

A roadblock for NTC is its higher sensitivity to process variations — i.e., the deviation of device parameters from their nominal values. Already in current-technology STC multicores, process variations result in noticeable differences in power and performance across the different cores of a chip [40]. At NTC, due to the low operating Vdd [39], the same amount of process variations causes a substantially larger impact on transistor speed and power consumption variations. Process variations are undesirable because they result in chips that consume more static power, cycle at lower frequencies, and can even be faulty.

Process variations should be addressed at multiple levels, including at the computer architecture level. To confront variations at the architecture level, we first need models of process variations and how they affect frequency and power, at a level of abstraction that is useful to microarchitects. Such models exist for STC (e.g., [41–45]). Unfortunately, none of them is applicable to

NTC — NTC uses new memory structures and requires new delay and power models.

This thesis presents the first microarchitectural model of process variations for NTC. The model, called *VARIUS-NTV*, extends the existing VARIUS variation model [45]. It models how variation affects the frequency attained and power consumed by cores and memories in an NTC manycore, and the timing and stability faults in SRAM cells at NTC. The key aspects include: (i) adopting a gate-delay model and an SRAM cell type that are tailored to NTC, (ii) modeling SRAM failure modes emerging at NTC, and (iii) accounting for the impact of leakage current in SRAM timing and stability models.

We evaluate a simulated 11nm, 288-core tiled manycore at both NTC and STC. Our results show that the expected process variations induce higher differences in frequency (f) and power at NTC than at STC. For example, the maximum difference in tile f within a chip is ≈3.7x at NTC and only ≈2.3x at STC. We evaluate different core-tiling organizations in the chip and different configurations of on-chip Vdd- and f-domains. Our experiments show that variation management is especially important at NTC. Finally, we validate our model against an experimental 80-core prototype chip [40].

In the following, Section 3.2 provides a background; Section 3.3 presents VARIUS-NTV variation model; Section 3.4 describes the manycore architecture evaluated; Sections 3.5 and 3.6 evaluate VARIUS-NTV for the architecture; Section 3.7 outlines the initial validation of VARIUS-NTV; and Section 3.8 discusses related work.

3.2 Background

3.2.1 Near-Threshold Computing (NTC) Basics

NTC refers to an environment where Vdd is set to a value only slightly higher than the transistors' Vth [37–39]. For current technologies, this roughly corresponds to Vdd ≈500mV, while the Vdd of conventional (or STC) environments is ≈1V.

NTC pushes back the manycore power wall by reducing the energy per operation several times compared to STC — at the expense of degrading the frequency of operation [38]. The result is that the power is expected to reduce by about an order of magnitude, allowing more cores to operate simultaneously for the same manycore power envelope. If the application has parallelism, this is a major advantage.

Figure 3.1 compares the scaling of three parameters under NTC, STC, and as imposed by classical CMOS theory [1]: supply voltage, transistor delay and power density. The X axis shows gate length to characterize each technology generation. Classical scaling relies on scaling Vdd down at every technology generation by a constant scaling factor κ. Both Vdd and transistor delay reduce at each generation, giving rise to a constant power density. Conventional STC scaling deviates from classical scaling in that the decrease of the transistor's Vth has practically stopped to keep subthreshold leakage under control, which in turn has prevented Vdd from scaling [2]. A consequence of this fact is that power density now keeps increasing. As we go from STC to NTC scaling, the curves experience vertical shifts. Specifically, as Vdd decreases (Figure 3.1(a)), power density goes down and transistor delay increases (Figure 3.1(b)).

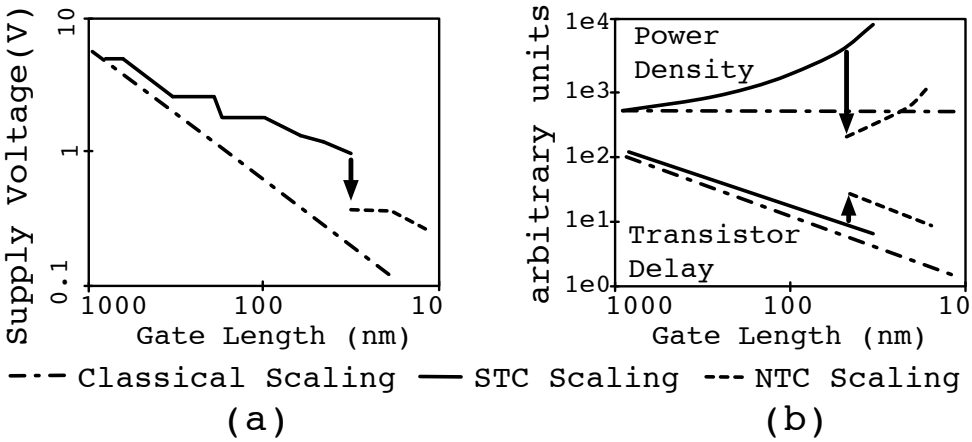


Figure 3.1: Parameter scaling under three scenarios [37].

In terms of energy and delay, NTC is close to a sweet spot. Figure 3.2 shows the inverse of energy per operation (labeled as energy efficiency) in MIPS/Watt (left Y axis) and the transistor delay (right Y axis) as a function of Vdd. In the NTC region, the energy efficiency is high and the transistor delay is relatively low. Away from this region, higher Vdd quickly results in substantially lower energy efficiency. Lower Vdd, on the other hand, quickly results in slower transistors.

3.2.2 The Impact of Process Variations at NTC

Each technology generation becomes increasingly vulnerable to process variations, which manifest across the chip as static, spatial fluctuations in transistor parameters around the nominal values [6, 46]. Within-die (WID) process variations are caused by systematic effects (e.g., due to

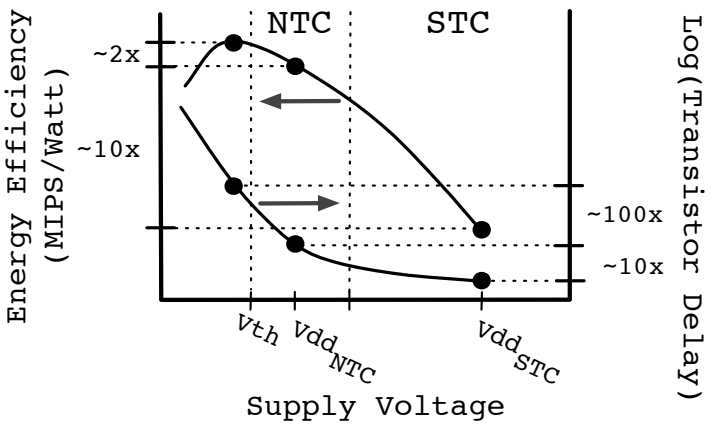


Figure 3.2: Impact of Vdd on energy efficiency and delay [38].

lithographic irregularities) and random effects (e.g., due to varying dopant concentrations) [47]. Two key process parameters affected by variations are Vth and the effective channel length (L_{eff}). The higher the Vth and L_{eff} variations are, the higher the variations in transistor switching speed and static power consumption are. This results in chips with increased variation in frequency and power consumption across cores and memories. Note that, in an environment with variation, the average core has lower frequency than before. This is because the slower transistors determine the frequency of the whole core. Moreover, the average core consumes more static power. The reason is that low-Vth transistors consume more additional power than high-Vth ones save.

Unfortunately, transistor delay and power consumption are more sensitive to variations in Vth and L_{eff} at NTC than at STC. Consider transistor delay first. At low Vdd, transistor delay is experimentally found to be more sensitive to changes in Vth [48]. For example, Figure 3.3 shows the transistor delay from the model of Markovic *et al.* [39] as Vth varies. For Vdd=0.6V, the difference in delay between transistors of Vth=0.25V and 0.35V is around 30ps, while for Vdd=0.4V, it jumps to over 200ps.

Dynamic power is also more sensitive to process variations at NTC than at STC. The reason is that dynamic power depends on the frequency and, as we have seen, at low Vdd, transistor delay (and hence frequency) is more sensitive to changes in Vth.

3.2.3 Modeling Process Variations at STC: VARIUS

There are several microarchitectural models that analyze the impact of process variations on processors and memories at a level that is useful to microarchitects (e.g., [41–45]). However, these

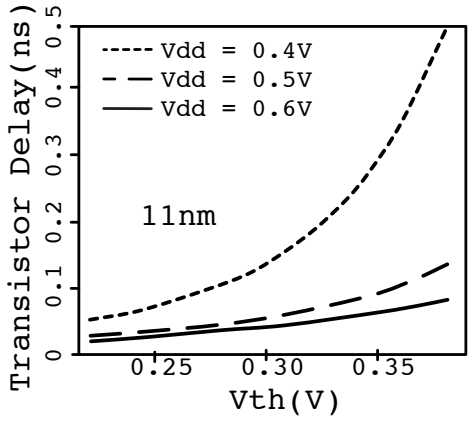


Figure 3.3: Transistor delay for different Vth.

works only apply to STC, and not to NTC. In this thesis, we take one of these models, namely VARIUS [45], and substantially extend it so that it applies to NTC. To understand our contributions, we briefly describe VARIUS.

VARIUS models variations in Vth and L_{eff}. It models their systematic component by dividing the die into a grid and assigning to each grid point a ΔVth and ΔL_{eff} value as sampled from a multivariate normal distribution with μ=0 and σ_{sys}. Moreover, these values have a spatial correlation that follows a spherical function. With this function, the correlation between two points only depends on their Euclidean distance. At a distance equal to zero, the correlation is one. The correlation then decreases with distance and, at a distance called *Correlation Range* (ϕ), the correlation becomes zero. VARIUS models the random component of variation with a normal distribution with μ=0 and σ_{ran}.

VARIUS plugs the Vth and L_{eff} variations in the alpha-power law (Equation 3.1) and in the equation for static power [49]. It then finds the variation in transistor (and gate) delay and transistor static power, respectively.

$$t_g \propto \frac{Vdd \times L_{eff}}{\mu(Vdd - Vth)^{\alpha}} \tag{3.1}$$

To find the distribution of delay of a pipeline stage, VARIUS proceeds differently depending on whether the stage has only logic, only an SRAM memory access, or a combination of both. For logic, it assumes that wire delays do not suffer from variations and, knowing the number of

gates in a logic path, it uses the gate delay variation computed above to compute the path delay variation. If VARIUS knows the distribution of the logic path delays in the stage (e.g., from Razor data [50]), it can estimate the distribution of variation-afflicted logic path delays.

For a stage with a memory access, VARIUS models the 6-transistor SRAM cell of Figure 3.4(a). Using the variation in transistor delay, it computes the variation in cell read access time. It assumes that the read access time is more critical than the write access time. Then, using the cell access time, it computes the memory line access time. Note that the pipeline stage also contains some logic, namely the decoder, the logic at the intersection of word- and bit-line, and the logic at the sense amplifier. The delay through all this logic is modeled using the previous logic-stage model and is added to the memory access delay to find the distribution of total path delay in the stage.

For pipeline stages that combine both logic and memory access, VARIUS estimates the delay distribution by appropriately weighting the delay of a logic stage and a memory stage. Finally, the pipeline stage with the longest delays determines the safe frequency of the processor.

The static power (P_{STA}) in the processor (or memory module) is found by integrating the P_{STA} of all of its transistors. VARIUS uses statistical principles to find a normal distribution for the processor's P_{STA} as a function of the normal distributions of the transistors' P_{STA}.

3.3 VARIUS-NTV: A Microarchitectural Model of Process Variations for NTC

VARIUS-NTV builds on VARIUS [45] to develop a microarchitectural model of process variations and resulting timing errors that is valid at NTC. Much of the general approach that VARIUS uses still applies to NTC — although the values of most parameters change. However, there are several important aspects that require complete redesign. This is where VARIUS-NTV contributes.

The main contributions of VARIUS-NTV are in four dimensions, which address four major limitations of VARIUS: (i) the VARIUS model for gate delay is based on the alpha-power law, which is only accurate for Vdd much larger than Vth; (ii) the VARIUS memory model uses a 6-transistor SRAM cell, which cannot reliably operate at NTC; (iii) for SRAM cells, the VARIUS model only considers read access (or timing) failures, while other memory failure modes dominate at NTC; and (iv) in the SRAM failure analysis, VARIUS neglects the impact of leakage while, at NTC, the impact of leakage is substantial.

In this section, we present the main contributions of VARIUS-NTV.

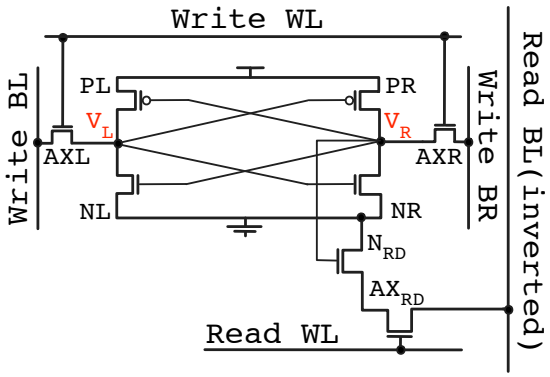


Figure 3.4: SRAM cell architecture: conventional 6-transistor cell (a) and 8-transistor cell (b). V_R and V_L are the voltages at the nodes indicated, which are referred to as nodes R and L, respectively.

3.3.1 Gate Delay

To model the gate delay (t_g), VARIUS uses the alpha-power law (Equation 3.1), where α is a process parameter capturing carrier velocity saturation, and μ identifies the carrier mobility as a function of the temperature (T). This equation does not model the NTC region accurately. There are alpha-power law variants [51–54] that attempt to extend the model to the subthreshold region. Usually, they come with an increased number of fitting parameters that have no direct physical interpretation. Furthermore, that they cover the subthreshold region does not necessarily imply that they model NTC properly.

Consequently, in VARIUS-NTV, we use the EKV-based [55] model proposed by Markovic *et al.* [39]. The formula for the on-current is given in Equation 3.2, where v_t is the thermal voltage and n a process-dependent parameter determined by subthreshold characteristics. The carrier mobility's T dependence is $\mu \propto T^{-1.5}$.

$$I \propto \mu / L_{eff} \times n \times v_t^2 \times ln^2(e^{\frac{Vgs-Vth}{2 \times n \times v_t}} + 1) \tag{3.2}$$

The resulting gate delay, obtained from CV/I, is shown in Equation 3.3. The equation captures the variation in gate delay as a function of the variation in Vth and L_{eff}. Since the EKV model covers all regions of operation, Equation 3.3 is equally valid at STC and NTC. In all cases, Vth is a function of Vdd and temperature as per Equation 3.4, where Vth_0, Vdd_0 and T_0 are the nominal values of these parameters, and k_T and k_{DIBL} represent constants of proportionality capturing the

impact of T and DIBL (Drain Induced Barrier Lowering) on Vth, respectively.

$$t_g \propto \frac{Vdd \times L_{eff}}{\mu \times n \times v_t^2 \times ln^2(e^{\frac{Vdd-Vth}{2 \times n \times v_t}} + 1)} \tag{3.3}$$

$$Vth = Vth_0 + k_{DIBL}(Vdd - Vdd_0) + k_T(T - T_0) \tag{3.4}$$

3.3.2 SRAM Cell

VARIUS uses the conventional 6-transistor cell shown in Figure 3.4(a). This cell requires careful sizing of the transistors, since it poses conflicting requirements on the AXR and AXL access transistors to prevent both read and write failures. While such a design is typical at STC, it becomes very hard to use at NTC, where transistors are more sensitive to process variations. One way to address this problem is to power SRAMs at a higher Vdd than the logic. Unfortunately, this approach is costly, since cache memory and logic blocks are often highly interleaved in the layout. Moreover, it requires extra voltage regulators in the platform, and results in additional design, validation, and testing issues. Finally, it is hardly scalable: as we move to smaller technologies, the relative difference between the safe SRAM and logic voltages increases, diminishing the power reduction benefit of NTC.

Consequently, VARIUS-NTV uses the 8-transistor cell of Figure 3.4(b) [56, 57]. This cell is easier to design reliably because it decouples the transistors used for reading (AX_{RD} and N_{RD}) and those for writing (the rest). Compared to the 6-transistor cell, read and write timing margins can be independently optimized with marginal increase in cell area [56]. In addition, of the five types of SRAM failure modes (read timing, read upset, write stability, write timing, and hold) [58], this cell eliminates read upset failures because the cell's internal nodes are decoupled from the read bit-line (BL).

3.3.3 Memory Failure Modes

While VARIUS only considers read timing failures, VARIUS-NTV models all of the SRAM failure modes (except read upsets, which cannot occur in the 8-transistor cell because a read cannot flip the cell contents by construction). We now describe how VARIUS-NTV models them.

Hold Failure

In a cell storing 0 ($V_R = 0,\ V_L = 1$), at low Vdd, the voltage V_L decreases by construction. This is because, when the cell is not accessed, although NL, PR, and the access transistors are off, there is leakage through NL and AXL. A hold failure occurs when the leakage current through the NL and AXL transistors in Figure 3.4(b) reduces V_L below the V_{SWITCH} of the $PR - NR$ inverter while the cell is not being accessed. At that point, the cell's state is lost.

To model these failures at a given Vdd, VARIUS-NTV uses Kirchhoff's current law to compute V_L and V_{SWITCH} at Vdd. V_L is extracted from $I_{PL}(V_L) - I_{NL}(V_L) - I_{AXL}(V_L) = 0$, where

$$\begin{aligned} I_{PL}(V_L) &\propto \mu/L_{eff} \times n \times v_t^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times v_t}} + 1) \\ I_{NL}(V_L) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times v_t}} \\ I_{AXL}(V_L) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times v_t}} \end{aligned} \tag{3.5}$$

and where Vth in each equation is expressed as a (different) function of V_L.

Similarly, V_{SWITCH} is extracted from $I_{PR}(V_{SWITCH}) - I_{NR}(V_{SWITCH}) + I_{AXR}(V_{SWITCH}) = 0$ for the $PR - NR$ inverter when V_{IN} = V_{OUT} [58].

The hold failure probability of a cell is $P_{Cell,Hold} = P[V_L(V_{dd}) - V_{SWITCH}(V_{dd}) < 0]$. Then, the hold failure probability of a line is $P_{Line,Hold} = 1 - (1 - P_{Cell,Hold})^{line\_size}$, where *line_size* is the number of cells per line, and $1 - (1 - P_{Cell,Hold})^{line\_size}$ gives the probability that at least one cell fails. A line is faulty if at least one of its cells is faulty. The failure probability of cells is assumed independent in this case.

To avoid hold failures, the minimum allowable supply voltage, $V_{ddMIN,Cell}$, is obtained by solving $V_L(V_{ddMIN,Cell}) = V_{SWITCH}(V_{ddMIN,Cell})$ under variation. Then, $V_{ddMIN,Line} = max(V_{ddMIN,Cell})$ for all the cells in the line.

Write Stability Failure

Without loss of generality, we focus on a cell that stores a 0 (V_R=0 and V_L=1). VARIUS-NTV computes the voltage (V_{LW}) that node L reaches when the write BL is set to 0 (where $BR = 1$) and the write duration is extended to infinity. If the value is above the switching threshold of the $PR - NR$ inverter (V_{SWITCH}), then a write failure occurs.

The V_{LW} distribution is computed using Kirchhoff's current law at node L, from $I_{PL}(V_{LW}) -$

$I_{NL}(V_{LW}) - I_{AXL}(V_{LW}) = 0$. The V_{SWITCH} distribution is extracted as explained above in the hold failure analysis.

In all cases, transistor parameters are subjected to the variation model. Finally, the per-cell probability of write stability failure becomes $P_{Cell,WStab} = P[V_{LW} - V_{SWITCH} > 0]$. A memory line suffers from write stability failure if there is at least one cell in the line suffering from it.

Read Timing Failure

VARIUS-NTV computes the random variable that captures the time taken to generate a detectable voltage drop on the read bit-line as

$$D_{VarReadCell} \propto \frac{1}{I_{AX_{RD}} + \sum I_{STA}} \tag{3.6}$$

where $I_{AX_{RD}}$ is the bit-line discharge current through the AX_{RD} transistor in Figure 3.4(b), and $\sum I_{STA}$ is the leakage over all of the cells attached to the bit-line. To calculate the distribution of $1/I_{AX_{RD}}$, first, the source voltage of AX_{RD}, V_{RD}, is extracted by solving the Kirchhoff's law at this node, from $I_{AX_{RD}}(V_{RD}) = I_{N_{RD}}(V_{RD})$. When reading from a cell storing 1 (V_R=1 and V_L=0), the transistor currents follow from:

$$\begin{aligned} I_{AX_{RD}} &\propto \mu/L_{eff} \times n \times v_t^2 \times ln^2(e^{\frac{Vdd - V_{RD} - Vth}{2 \times n \times v_t}} + 1) \\ I_{N_{RD}} &\propto \mu/L_{eff} \times n \times v_t^2 \times ln^2(e^{\frac{Vdd - Vth}{2 \times n \times v_t}} + 1) \end{aligned} \tag{3.7}$$

where Vth in each equation is expressed as a (different) function of V_{RD}.

Then, the probability distribution of $D_{VarReadCell}$ can be attained by applying those of Vth and L_{eff} given by the variation model to $\frac{1}{I_{AX_{RD}(V_{RD})} + \sum I_{STA}}$. Following the VARIUS methodology, the maximum of $D_{VarReadCell}$ over all of the cells in a line is the time to read an entire memory line $D_{VarReadLine}$. Finally, the probability of read access failure ($P_{ReadAccess}$) is $P[D_{VarReadLine} > t_{READ}]$, where t_{READ} is the designated read duration.

Write Timing Failure

Given a cell without write stability failure, VARIUS-NTV models a write timing failure by computing $D_{VarWriteCell}$. This is the time that node L takes to reach the switching threshold (V_{SWITCH})

of the *PR-NR* inverter. It is:

$$D_{VarWriteCell} \propto \frac{1}{I_L} = \int_{Vdd}^{V_{SWITCH}} dv_L / i_L(v_L)$$
$$i_L(v_L) = i_{PL}(v_L) - i_{NL}(v_L) - i_{AXL}(v_L) \tag{3.8}$$

where I_L is the discharge current at node L during the write, obtained following [58]. $i_L(v_L)$ is a function of Gaussian random variables Vth and L_{eff} under process variation. It is obtained with Kirchhoff's current law.

After obtaining the probability distribution for $D_{VarWriteCell}$, we compute the distribution of the maximum of $D_{VarWriteCell}$ over all of the cells in a line. Finally, the probability of write timing failure ($P_{WriteTiming}$) is $P[D_{VarWriteLine} > t_{WRITE}]$, where t_{WRITE} is the designated write duration.

3.3.4 Impact of Leakage

At NTC, the magnitude of the leakage current (I_{off}), decreases when compared to STC. However, the on-current (I_{on}), decreases even more due to lower Vdd. Hence, the relative impact of I_{off} increases. Consequently, unlike VARIUS, VARIUS-NTV takes into account the impact of the leakage current on SRAM timing and stability, as we have seen in previous sections. As part of I_{off}, we only consider subthreshold leakage; we exclude gate leakage because we assume high-K metal gate devices like the ones currently in use.

3.4 Manycore Architecture Modeled

To evaluate VARIUS-NTV, we model an 11nm manycore architecture that operates at NTC. The manycore is organized in tiles (36 in our default configuration) for ease of design (Figure 4.1). Each tile has a tile memory and several cores (8 in our default configuration), each with a per-core memory. Each core is a single-issue engine where memory accesses can be overlapped with each other and with computation. Each tile memory is a bank of a shared L2 cache, while the per-core memories are L1 caches. Data in the L1 caches is kept coherent with a directory-based MESI coherence protocol where each pointer corresponds to one tile. The cores are connected with a bus inside each tile and with a 2D torus across tiles. Table 4.2 shows the default architecture and technology parameters. In the table, all of the parameters that are not labeled with STC refer to

the NTC environment.

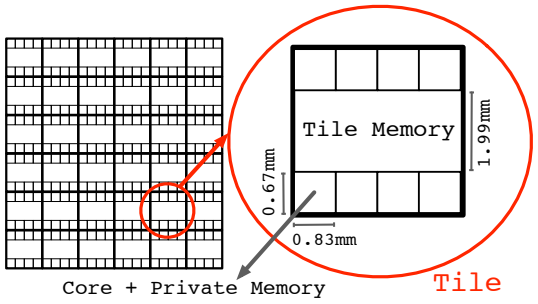


Figure 3.5: Manycore architecture used to evaluate VARIUS-NTV.

We evaluate an STC version of the manycore and three NTC versions of it. The three NTC versions differ based on the use of voltage and frequency domains, as listed in Table 3.2.

System Parameters	
Technology node: 11nm Num. Cores: 288 Num. Tiles: 36 (8 cores/tile)	P_{MAX} = 100W $T_{MAX} = 80^{o}$C Chip area ≈ 20mm x 20mm
Variation Parameters	
Correlation range: $\phi = 0.1$ Total $(\sigma/\mu)_{Vth} = 20\%$; equal contrib. systematic & random	Sample size: 100 chips Total $(\sigma/\mu)_{Leff} = 10\%$; equal contrib. systematic & random
Technology Parameters	
Vdd_{NOM} at STC = 0.77V Vth_{NOM} at STC = 0.30V f_{NOM} at STC = 3.3GHz $f_{interconnect}$ at STC = 2.5GHz $k_T = -1.5mV/K$; $n = 1.5$	Vdd_{NOM} at NTC = 0.55V Vth_{NOM} at NTC = 0.33V f_{NOM} at NTC = 1.0GHz $f_{interconnect}$ at NTC = 0.8GHz $k_{DIBL} = -150mV/V$
Architectural Parameters	
Per-core memory: 64KB WT, 4-way, 2ns access, 64B line On-chip network: bus inside tile and 2D-torus across tiles Crossing a f domain boundary: 2ns	Tile memory: 2MB WB, 16-way, 10ns access, 64B line Directory-based MESI Avg. memory round-trip access time (before contention): ≈80ns

Table 3.1: Architecture and technology parameters.

The technology parameters used in Table 4.2 are derived from ITRS [59] and projected trends from industry. Every single experiment is repeated for 100 chips with different variation profiles, and we present the average. More samples beyond 100 do not change the results noticeably.

Name	NTC Manycore Configuration
MVMF	Multiple V_{dd} and multiple f domains (one per tile).
SVMF	Single chip-wide V_{dd} domain and one f domain per tile.
SVSF	Single chip-wide V_{dd} and f domains.

Table 3.2: Configurations for the NTC manycore.

3.5 Experimental Setup

We evaluate VARIUS-NTV by using it to estimate the performance and power consumption of the manycore architecture of Section 3.4. We interface Pin [60] over a user-level pthreads library to the SESC [33] cycle-level architectural simulator. SESC estimates both execution time and energy consumed. The energy analysis relies on McPAT [61] scaled to 11nm. An updated version of HotSpot takes the detailed layout of the chip and models the temperature, in turn affecting the leakage energy in a feedback loop. VARIUS-NTV is implemented in R [62].

In our experiments, we run multi-programmed workloads that contain some or all of the following 8 PARSEC applications: blackscholes, ferret, fluidanimate, raytrace, swaptions, canneal, dedup, and streamcluster. Each application can run with 4, 8 or 16 threads. For each application, we measure the complete parallel section (called Region of Interest or ROI) running the *simsmall* input data set.

3.6 Evaluation

In our evaluation, we first describe how we set the operating voltages and frequencies of the manycore, then assess the impact of process variations in NTC and STC environments, and then explore some design parameters.

3.6.1 Computing the Operating Point

To determine the operating Vdd and f at NTC, our model starts with SRAM blocks. Our goal is to estimate Vdd_{MIN}, the minimum sustainable Vdd. It is set by hold and write stability failure analyses.

Our model first finds the minimum Vdd needed to avoid hold failures, namely Vdd_{hold}. The Vdd_{hold} distribution is attained by solving $V_L(V_{dd,hold}) = V_{SWITCH}(V_{dd,hold})$, where the former is the voltage at node L (Figure 3.4(b)), while the latter is the switching threshold of the *PR-NR* inverter.

The chosen Vdd_{hold} value is obtained at the 3σ of the distribution — after approximating to a normal distribution. Our model then proceeds with write stability failure analysis, to guarantee that the chosen Vdd_{hold} also avoids write stability failures. At this step, a higher Vdd may emerge, if the write stability failure rate at Vdd_{hold} remains higher than the target tolerable error rate. The resulting Vdd is Vdd_{MIN}.

Once Vdd_{MIN} is picked, VARIUS-NTV considers timing issues in order to set the f. The selected f is determined by the slowest component of the chip, based on our model's analysis of path delay distributions at Vdd_{MIN}. For logic blocks, the analysis follows that of VARIUS [45]. For SRAMs, it can be shown that, for the parameters considered, write timing requires longer delays than read timing for the same Vdd. This is consistent with the work of Abella *et al.* [63]. Hence, write timing analysis determines the path delays in each SRAM block. To determine the maximum path delay, VARIUS-NTV approximates the path delay distributions to normal ones and picks the 3σ cut-off point. This maximum delay determines the f at Vdd_{MIN}.

3.6.2 Impact of Process Variations at NTC and STC

To examine the impact of WID process variations on the f and power consumption at NTC and STC, we consider three types of on-chip blocks separately: logic (the core pipelines), small memories (the per-core local memories) and large memories (the per-tile memories). We do this because they have different critical path distributions. In all cases, the f for a block is determined by finding the distribution of the path delays in the block at Vdd_{NOM} and then picking, as the period for the block, the delay at the 3σ of the distribution. The power of the block is the sum of the static and dynamic components.

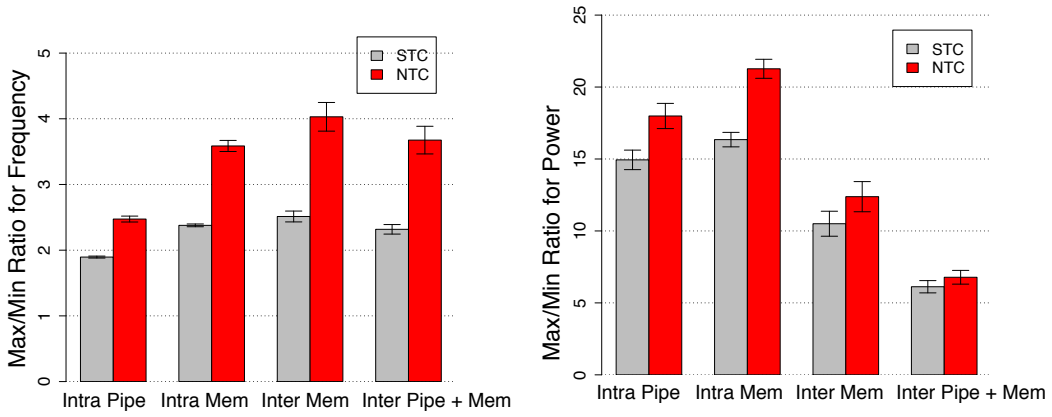
We consider intra-tile variations first. In each tile, we compute the ratio of the frequencies of the fastest and slowest pipelines in the tile. We then take the average of the ratios across all tiles (*Intra Pipe*). We repeat the same process for local memories in the tile to calculate *Intra Mem*. Finally, for the power consumption, we take the power ratio of highest to lowest consuming pipelines, and highest to lowest consuming local memories, to compute *Intra Pipe* and *Intra Mem*, respectively.

For inter-tile variations, we measure the ratio of the frequencies of the fastest and slowest tile memories on chip (*Inter Mem*). We then consider the frequency that each tile can support (the lowest frequency of its pipelines, local memories and tile memory), and compute the ratio of the frequencies of the fastest and slowest tiles (*Inter Pipe+Mem*). Finally, we repeat the computations

for power (*Inter Mem* and *Inter Pipe+Mem*). We report the mean of the experiments for 100 chips.

Figure 3.6 compares these ratios for NTC and STC. Figure 3.6(a) shows the f ratios. We observe that the frequency ratio of the fastest to the slowest blocks is substantially higher at NTC than at STC — for the same process variation profile. For example, *Inter Pipe+Mem* at NTC is 3.7, while it is only 2.3 at STC (Figure 3.6(a)). This is because a low Vdd amplifies the effect of process variations on delay.

Figure 3.6(b) shows the power ratios. The variation in total power also increases at NTC. However, the relative difference in power ratios between NTC and STC is generally smaller than the relative difference in frequency ratios. The reason is that power includes both dynamic and static power, and the ratios for static power are the same for NTC and STC. Consequently, the relative difference in power ratios is smaller. Still, the absolute difference is significant. Consequently, the chip is more heterogeneous at NTC.



(a) Frequency (b) Power

Figure 3.6: Impact of variations at NTC and STC.

These experiments have used a fixed, safe Vdd_{NOM} for the whole chip. In reality, process variations in the SRAM cells result in each tile supporting a different Vdd_{MIN}, the minimum sustainable Vdd to avoid failures. Such Vdd_{MIN} values are lower than Vdd_{NOM} for many tiles. Figure 3.7 shows the distribution of the Vdd_{MIN} values for all the tiles in a sample chip at NTC. The data is shown as a histogram. We can see that the Vdd_{MIN} values of tiles in a chip vary along a significant 0.46-0.58V range.

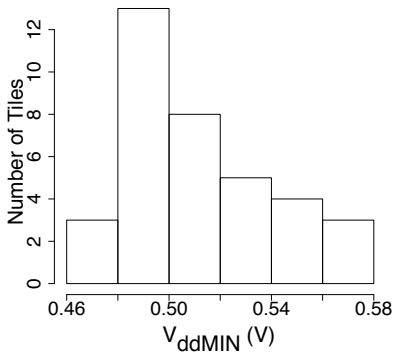


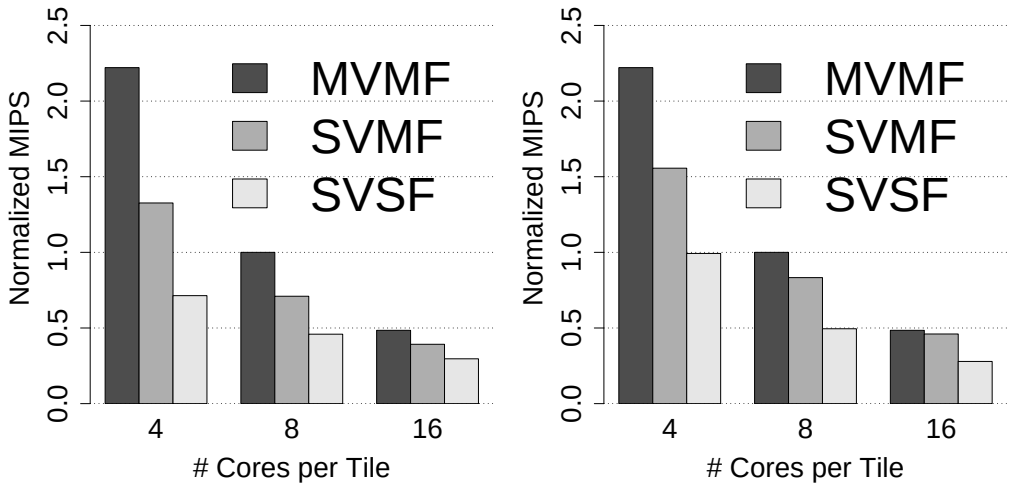
Figure 3.7: Values of Vdd_{MIN} for all the tiles of a representative chip at NTC.

3.6.3 Design Space Exploration

A promising way to combat the increased impact of process variations is to rely on fine grain, per-tile Vdd and f tuning. To quantify the effect, we compare the manycore configurations of Table 3.2 across different tile granularities ranging from 4 cores per tile to 16 cores per tile. *MVMF* is an environment with a Vdd and an f domain per tile; *SVMF* has a single Vdd domain in the chip but one f domain per tile; finally, *SVSF* characterizes a variation-oblivious environment, with a single Vdd and f domain per chip.

Figure 3.8 compares the performance (in normalized *MIPS*) of our 288-core NTC chip for the different environments. We consider two workload scenarios: one where we use all the tiles in the chip (Figure 3.8(a)) and one where we only use about half of the tiles (Figure 3.8(b)). Specifically, we use 128 out of the 288 cores and leave the others idle. Figure 3.9 repeats the analysis for STC.

In each figure, we keep the total number of cores in the chip constant, and perform a sensitivity analysis of different tile granularities: 4, 8 or 16 cores per tile. In each case, the workload consists of 4-threaded, 8-threaded, or 16-threaded parallel applications, respectively, from PARSEC. Each application uses one tile, and we report the average performance of the workload in *MIPS*. In each plot, to make the comparison fair, the power consumed by all of the environments is kept constant. In *MVMF*, the per-domain Vdd and f are set as per Section 3.6.1. Specifically, each tile runs at the tile-specific Vdd_{MIN}, and at the maximum f that it can support at this voltage. In *SVMF*, all the tiles in the chip run at the maximum of the Vdd_{MIN}s across all tiles. The per-tile frequencies are increased accordingly. Finally, in *SVSF*, the chip uses the same voltage as *SFMV* but it runs at the chip-wide minimum of per-tile frequencies. Recall that the Vdd_{MIN} of a tile represents the maximum Vdd_{MIN} across its components, where the f of a tile corresponds to the



(a) 100% Use (b) ≈ 50% Use

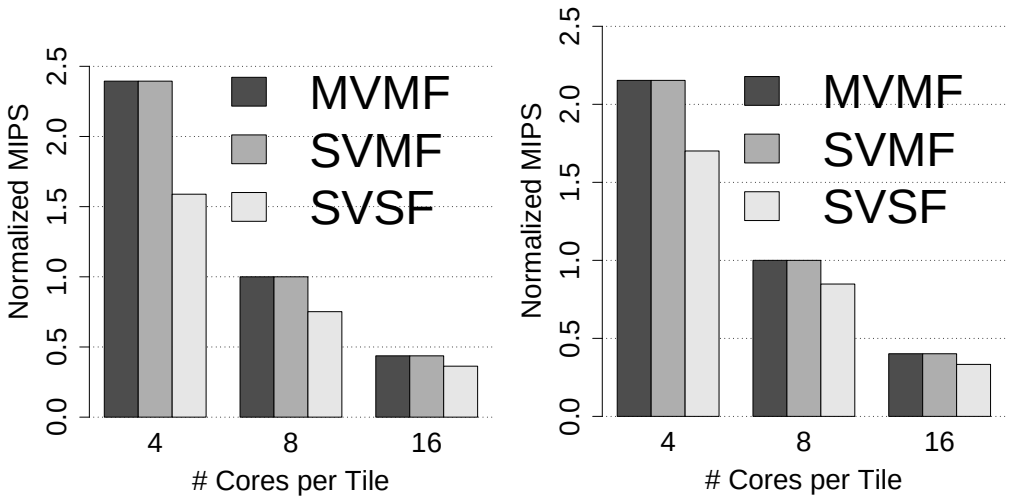
Figure 3.8: Performance of our 288-core chip at NTC with different tile sizes and configurations. The charts correspond to using all the tiles (a) and using approximately only half (b).

minimum *f* across its components at the designated tile Vdd. The applications are assigned to tiles according to highest average IPC application to highest *f* tile. After the *MIPS* of each environment is computed, it is normalized to that of *MVMF* for an 8-core tile in each plot.

Starting with the fully-utilized chip (Figure 3.8(a)), we observe that *SVMF* only attains 59%, 71%, and 81% *MIPS* of *MVMF*, for 4-core, 8-core, and 16-core tiles, respectively. This is because it does not exploit the multiple Vdd domains of *MVMF*. The difference between the two bars gets larger as the tile granularity becomes finer, as *MVMF* tracks core-to-core variations closer. *SVSF* in this case only reaches 32%, 46%, and 61% *MIPS* of *MVMF*, for 4-core, 8-core, and 16-core tiles, respectively. As the tile granularity increases, the differences between the different configurations diminish.

Figure 3.8(b) repeats the experiment when only ≈ half of the tiles are busy. For *MVMF*, we pick the 32, 16, and 8 most *MIPS/W*-efficient tiles for 4-, 8-, and 16-cores per tile granularity, respectively, and then assign the applications of higher IPC to the faster tiles in turn. The resulting power consumption is the power budget that we allow to the other environments. The other environments pick their 32, 16 or 8 most *MIPS/W*-efficient tiles that satisfy the budget. We see similar trends as in Figure 3.8(a) except that the drop in *MIPS* is not as large. The reason is that each environment now picks a subset of energy-efficient tiles — leaving energy-inefficient ones idle.

Finally, in Figure 3.9, the experiments are repeated for STC. For STC, *MVMF* and *SVMF* become



(a) 100% Use (b) ≈ 50% Use

Figure 3.9: Performance of our 288-core chip at STC with different tile sizes and configurations. The charts correspond to using all the tiles (a) and using approximately only half (b).

equivalent, since the nominal STC Vdd is high enough to produce a safe operating point across all of the tiles. There is no need to set the Vdd of some tiles higher or lower depending on their Vdd_{MIN}. Apart from this, while generally the same trends apply as under NTC operation, the MIPS loss as incurred by *SVSF* operation is much less.

3.7 Model Validation

Our initial validation of VARIUS-NTV involves a validation of the parameters used and a comparison to the results reported in an experimental chip.

Validation of Model Parameters

VARIUS-NTV builds on the VARIUS variation and timing error model which, as explained in [45], was calibrated with experimental data from Friedberg *et al.* [64] and Razor [50], and validated with error rates in logic and memory [45]. To validate the new VARIUS-NTV formulas, we start with Vth, which is a complex function of Vdd, L_{eff}, and other technology parameters. We obtained a version of the 12nm Predictive Technology Model (PTM) from Yu Cao from Arizona State University [65]. We compared the Vth values generated by VARIUS-NTV to those generated by the BSIM analytical model [66], and HSPICE. The Vth values from VARIUS-NTV closely track those

from both HSPICE and BSIM with less than 1% error over the designated Vdd range. The main source of discrepancy is the accuracy of modeling the DIBL effect.

We then used Vth values from VARIUS-NTV to generate values for gate delay and static power. We compared the values to HSPICE measurements of a FO4 inverter chain. The delay and static power scaling trends of VARIUS-NTV follow HSPICE within a 10% of error for our Vdd range.

Comparison to Silicon Measurements

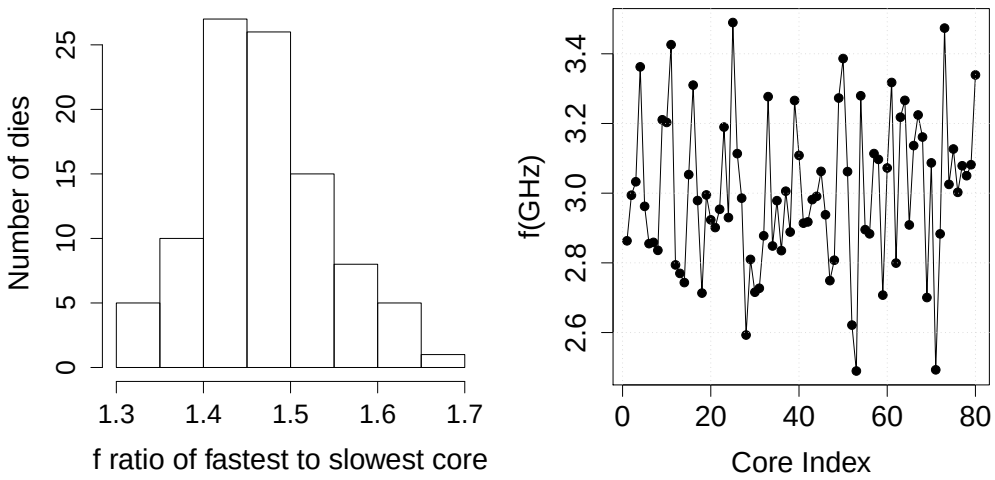
To further validate VARIUS-NTV, we compare its outputs to the variation measurements from Intel's 80-Core TeraFLOPS processor [40]. To this end, we experimented with a 12mm$\times$20mm chip that mimicks the TeraFLOPS processor, where each core (which they call tile) has 2 floating point units, a 3KB instruction memory, and a 2KB data memory. According to the chip micrograph, the chip organizes the 80 cores into 10 rows and 8 columns. To match their technology parameters, we adapted VARIUS-NTV to a 65nm CMOS technology with a Vdd_{NOM} of 1.2V.

Figure 8 in [40] depicts the *measured* variation in core frequency (f_{MAX}) for the 80 cores of a single die at 50^{o}C and Vdd=0.8V. At 0.8V, the authors report a ratio of highest core frequency to lowest core frequency equal to 1.62.

We repeat the conditions in which these measurements were taken to the extent that we can. We generate VARIUS-NTV frequency maps for 100 sample dies, assuming $(\sigma/\mu)_{Vth} = 5\%$ for the 65nm technology, with an equal contribution of random and systematic variation. The histogram of the resulting ratios of highest core frequency to lowest core frequency as generated by VARIUS-NTV is shown in Figure 3.10(a). As shown in the histogram, VARIUS-NTV produces an average value of $\approx$1.48 for the ratio of frequencies, with a 95% confidence interval of (1.452, 1.483).

Further, Figure 3.10(b) shows the frequency distribution of the cores in one of the dies, as generated by VARIUS-NTV at 0.8V. For this particular die, the ratio of highest core frequency to lowest core frequency is $\approx$1.4. This figure is very similar to Figure 8 in [40].

Recall that the 80-Core processor does not represent an NTC design. However, our validation experiments are run at the relatively low 0.8V (where the nominal Vdd at 65nm is 1.2V). No further measured data is provided in [40] below 0.8V. To our knowledge, there is no detailed variation characterization of any NTC chip that is available.



(a) (b)

Figure 3.10: Data generated by VARIUS-NTV that replicates the data presented in [40]. Chart (a) shows a histogram of the ratios of highest core frequency to lowest core frequency over 100 dies. Chart (b) shows the frequency map for one of the sample dies.

3.8 Related Work

There are several microarchitectural models that analyze the impact of process variations on the frequency and power of processors and memories at a level that is useful to microarchitects. They include the work of Humenay *et al.* [41], Liang and Brooks [42], Marculescu and Talpes [43], Romanescu *et al.* [44], and Sarangi *et al.* [45] (on which this work builds) among others. As indicated before, these works only apply to STC, and not to NTC.

A few papers include a good description of the challenges and issues at NTC [37–39].

There are many other works that are related to evaluating the impact of process variation, mostly in STC environments. We list some of the most relevant here. Humenay *et al.* demonstrate that WID process variations lead to considerable performance and power consumption asymmetry among the cores in a CMP [41]. To minimize such asymmetry, they propose per-core ABB and ASV. Donald and Martonosi analyze core-to-core power variations in a CMP due to WID variation [67]. They propose to turn off cores when they consume excessive leakage power in order to maximize the chip-wide performance/power. Herbert and Marculescu examine the impact of core size on the throughput of a fixed area chip in the presence of WID variations [68]. They find that smaller cores (thus more cores per chip) running at independent f lead to higher throughput than larger ones. Li and Martinez propose to optimize the number of active cores and their

*Vdd*s and *f*s jointly while running a workload on a CMP [69] where they apply DVFS chip-wide rather than independently per core. In [70], Rangan *et al.* propose a throughput driven scheduling scheme to guarantee that a variation-afflicted chip performs very close to a perfect chip operating at the average frequency of the former. Rotem *et al.* [71] analyze the impact of single and multiple voltage and frequency domains in a CMP environment, considering power delivery limitations. They propose a clustered topology to maximize performance. The authors ignore the impact of variation. Finally, Teodorescu and Torrellas [32] examine the impact of process scheduling in the context of a manycore with variation. They provide heuristics to schedule the workload for performance or for power efficiency. It would be interesting to reproduce these works in the context of NTC.

3.9 Summary

To help confront process variations at the architecture level at NTC, we presented the first microarchitectural model of process variations for NTC. The model, called *VARIUS-NTV*, extends an existing variation model for STC. It models how variation affects the frequency attained and power consumed by cores and memories in an NTC manycore, and the timing and stability faults in SRAM cells at NTC. The key aspects include: (i) adopting a gate-delay model and an SRAM cell type that are tailored to NTC, (ii) modeling SRAM failure modes emerging at NTC, and (iii) accounting for the impact of leakage in SRAM failure models.

We evaluated a simulated 11nm manycore at both NTC and STC. Our results showed that the expected process variations induce higher differences in f and power at NTC than at STC. For example, the maximum difference in tile f within a chip is $\approx$3.7x at NTC and only $\approx$2.3x at STC. We evaluated different core-tiling organizations in the chip and different configurations of on-chip Vdd- and f-domains. Our experiments showed that variation management is more crucial at NTC. Finally, we validated our model against an experimental 80-core prototype chip.

4 ESCHEWING MULTIPLE V_{DD} DOMAINS AT NTC

4.1 Introduction

Manycore scaling now faces a Power Wall. Successive technology generations have been increasing chip power density, resulting in a situation where more cores can now be placed on a chip than can be concurrently operating. We urgently need new ways to execute more power- and energy-efficiently.

One way to attain energy-efficient execution is to reduce the supply voltage (Vdd) to a value only a bit higher than a transistor's threshold voltage (Vth). This environment is called Near-Threshold Voltage Computing (NTC) [37–39] — as opposed to the conventional Super-Threshold Voltage Computing (STC). Vdd is a powerful knob because it has a strong impact on both dynamic and static energy. According to initial data [37, 38], NTC can decrease the energy per operation by several times over STC. One drawback is a degradation in frequency, which may be tolerable through more parallelism in the application. Still, since many more cores can be executing concurrently within the chip's power envelope, the result is a higher throughput for parallel codes.

A major roadblock for NTC is process variations, namely the deviation of device parameters from their nominal specifications. Already at STC, a chip has regions with different speed and power consumption. At NTC, the same amount of process variations causes larger changes in transistor speed and power due to the low Vdd [39].

It is important to find techniques to cope with process variations in future NTC chips. Possibly, general approaches currently used to handle variations could be applied. They include, among other techniques, Adaptive Body Biasing (ABB) and, especially Vdd tuning (in the form of Adaptive Supply Voltage, ASV, or Dynamic Voltage Scaling, DVS), where the effectiveness increases with support for multiple finer grain Vdd and frequency (f) domains. With fine grain Vdd and f domains, we can separately adapt to the parameter values in the different regions of the chip.

Unfortunately, simply applying these techniques will not work at NTC. There is consensus that ABB will likely be ineffective in new technologies. Most importantly, supporting many on-chip

Vdd domains with conventional on-chip Vdd regulators is not power-efficient — hence hardly compatible with a highly energy-efficient environment such as NTC. The reason is that on-chip regulators — the only type we currently know how to build to provide the desired functionality — have a typical power efficiency of 70-90% [72,73]. Moreover, given the many cores that an NTC chip can include, many such regulators would be needed, multiplying the cost.

In this thesis, we examine how significant the problem of power-inefficient Vdd domains is at NTC. We also explore a potential alternative that tackles variation with a single chip-wide Vdd domain and multiple f domains. We call this new alternative *Polyomino*. Having f domains introduces an execution overhead (when a boundary is crossed) but not a significant power one. However, for such an approach to be competitive, we need to address two issues. The first one is to provide core-to-job assignment algorithms that deliver high performance per watt without relying on multiple Vdd domains. The second is to support chip-wide DVFS that adapts Vdd in intervals short enough to be competitive with an environment with on-chip Vdd regulators.

First, we show that, at NTC, a Polyomino chip delivers higher performance per watt than one with multiple on-chip Vdd domains supported by on-chip Vdd regulators. The reasons are: (i) the regulators' power inefficiencies, (ii) the increased Vdd guardband induced by fine grain Vdd domains (to handle deeper Vdd droops due to lower capacitance per Vdd domain), and (iii) the practical fact that any Vdd domain still has to include several cores. Second, we introduces core assignment algorithms for the Polyomino architecture that deliver high performance per watt while being simple. Finally, we show that the lower speed of Vdd changes during DVFS without on-chip Vdd regulators is perfectly tolerable.

In the following, Section 4.2 discusses multiple Vdd domains at NTC and presents Polyomino; Sections 4.3 and 4.4 discuss core assignment and fine-grain DVFS for Polyomino; Sections 4.5 and 4.6 evaluate the ideas; Section 4.7 presents a discussion; and Section 4.8 covers related work.

4.2 Eschewing Multiple Vdd Domains at NTC

Having multiple on-chip Vdd domains with independent voltage scaling can increase the energy efficiency of a manycore — e.g., by executing non latency-critical applications in low-Vdd domains. Since NTC is an energy-conscious environment, such support appears to suit it. In addition, NTC has two additional reasons to benefit from multiple Vdd domains. First, since WID process variations are more significant at NTC, chip neighborhoods at NTC are expected to benefit

more from the decoupling of Vdd values across the chip. The second reason is that, at NTC, Vdd is a particularly strong lever to affect the power and performance conditions of core operation. Specifically, small changes in Vdd have a relatively large impact on the performance and energy consumption of a core.

However, we uncover several limitations that make the use of multiple Vdd domains at NTC less attractive. The presence of such limitations suggests a different type of manycore architecture at NTC, and two challenges that need to be overcome for cost-effective operation. Next, we describe these limitations, the architecture, and the challenges.

4.2.1 Limitations of Multiple Vdd Domains at NTC

There are several effects that limit the cost-effectiveness of multiple on-chip Vdd domains in a power-conscious environment such as a future NTC chip (Table 4.1). The first one is the power loss in conventional on-chip Vdd regulators. Having Vdd regulators on chip is likely the only realistic way to support many domains — utilizing many off-chip regulators (e.g., 36) is too expensive. Unfortunately, such regulators have typical power efficiencies of only 70-90%, be they switching or low-dropout (LDO) regulators. For example, Ghasemi *et al.* [72] discuss LDO regulators with a range of efficiencies, while Kim *et al.* [73] discuss on-chip switching regulators that have a peak power efficiency of 77%. This is hardly compatible with a very power-conscious environment such as NTC.

Limitation	Reason
Power loss in on-chip Vdd regulators	Regulators have power efficiencies of 70-90%
Increased Vdd guardband to tolerate tolerate larger dynamic Vdd droops	Each domain has a lower capacitance than a large chip with a single Vdd domain
Inability to fully fine-tune Vdd for individual cores in one domain	To reduce cost and complexity, a domain still includes several cores

Table 4.1: Why multiple Vdd domains become less attractive in a future NTC environment.

The second effect is the likely need to increase the Vdd guardband in the finer grain Vdd domains, to tolerate more accentuated dynamic Vdd droops. Such deeper droops may be induced by the lower capacitance of a domain, compared to a large chip with a single Vdd domain. James *et al.* [74] discuss this problem for the IBM POWER6 processor. Increased Vdd guardbands imply lower power efficiencies.

Finally, there is the practical aspect that the low-power operation of NTC chips will result in

chips with many cores. Attempting to control Vdd on a per-core basis is expensive in hardware. Therefore, it is likely that each Vdd domain will include several cores. The differences between the cores in the same Vdd domain, as induced by the increased sensitivity to process variations, will lead to a Vdd setting that is suboptimal for individual cores — hence missing out on part of the potential gains.

4.2.2 An Alternative Approach for Future NTC: *Polyomino*

Since supporting multiple Vdd domains is costly at NTC, we propose an alternative manycore architecture. The architecture, called *Polyomino*, eschews multiple on-chip Vdd domains, for hardware simplicity and energy efficiency. It keeps a single Vdd across the whole chip. DVFS can be used, but it is applied globally across the chip.

To handle process variations more inexpensively than with fine grain Vdd and frequency (f) domains, Polyomino uses only f domains. Polyomino is organized in clusters of cores, where each cluster is a f domain. A cluster (domain) is characterized by the maximum frequency (f_{MAX}) that it can support at the lowest possible chip-wide safe voltage (Vdd_{NOM}). Vdd_{NOM} and the set of f_{MAX} are set at manufacturing-testing time. With many f domains, the chip has many degrees of freedom in an environment highly affected by process variation.

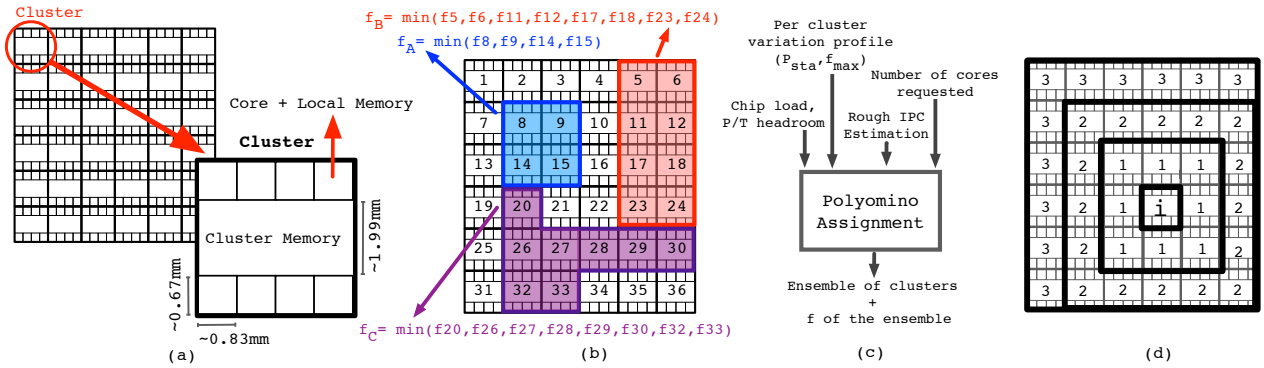


Figure 4.1: Example Polyomino architecture (a), its operation (b), the core assignment algorithm (c), and distance of clusters to cluster i (d).

Figure 4.1(a) shows a Polyomino manycore. This design has 6×6 clusters, where each cluster has a cluster memory and 8 cores with local memories.

To determine Vdd_{NOM} and the set of f_{MAX}, we proceed as follows. Each cluster's minimum sustainable Vdd, Vdd_{MIN} is set by the cluster's SRAM hold and write stability failure analyses to

ensure reliable operation. Then, the chip-wide Vdd_{NOM} becomes the maximum over all clusters' Vdd_{MIN}. After Vdd_{NOM} is set, timing tests in the SRAM and logic for each cluster i determine the maximum frequency f_{MAX_i} that the cluster can support at Vdd_{NOM}. Such frequency will be the default frequency of the cluster. It can be increased if Vdd increases over Vdd_{NOM}.

In Polyomino, where we give up multiple Vdd domains and, instead, rely on fine-grain f domains, two challenges appear: the need to (1) carefully control core-to-job assignment, and (2) effectively support fine-grain (i.e., short-interval) DVFS. We consider them next.

4.3 The Challenge of Core Assignment

4.3.1 Rationale: Simplicity and Effectiveness

Attaining energy-efficient performance in a manycore with hundreds of cores as enabled by NTC requires a good core-assignment algorithm. In such a manycore, the number of degrees of freedom in the assignment is vast. Hence, the challenge is to attain effective assignment while managing such complexity. Polyomino eases the complexity by eliminating Vdd domains: the assignment algorithm only needs to select the f of the chosen cores, rather than both Vdd and f. However, the inability to set multiple Vdds in the chip disables a potentially important knob for energy-efficiency. Hence, Polyomino's assignment algorithm needs careful design.

Recall that the chip is necessarily organized in clusters of cores to exploit WID variations and to enhance scalability. A cluster is the smallest f domain. Clocking all the cores in a cluster at the same f is reasonable, since the whole cluster is likely to have a similar value of the systematic component of process variations. In this environment, Polyomino simplifies core assignment further by assigning all the cores in a cluster as a group to a job. Any resulting unused cores in the cluster are power gated. Leaving them unused is typically not a problem because, in our environment, there is likely a surplus of cores. However, if cores are scarce, a cluster can take in multiple jobs.

A single parallel job may take multiple clusters. Such set of clusters is called an *Ensemble*. For assignment simplicity, an ensemble runs at a single f, which is equal to the lowest of the f_{MAX} of the constituting clusters. Moreover, running at a single f ensures that all the threads of the job make similar progress and typically results in faster overall execution [75] — especially in highly-synchronized codes.

When multiple jobs are running concurrently, each is assigned to a different ensemble, which

forms a separate f domain. An example is shown in Figure 4.1(b), where f_{MAX} for cluster i is depicted as f_i. Three ensembles are allocated to three jobs (A, B and C), giving rise to three independent f domains characterized by frequencies f_A, f_B and f_C. Each ensemble operates at the f of its slowest component cluster.

There are various design alternatives for the network that interconnects the clusters. One alternative is to have a single, separate f domain for the network [40]. This is a simple design, but requires that every single communication between clusters in an ensemble or between a cluster and memory crosses domains. An alternative is to break the network into multiple f domains — possibly including each router in the f domain of its neighbor cluster. This design would allow clusters within an ensemble to communicate without crossing f domains. However, longer-distance messages would still suffer. Moreover, the network is more complicated to design. Given that crossing a f domain only adds about 2 ns [76], we choose the simple, single f domain for the network. Hence, Polyomino's design does not eliminate f-domain crossings when clusters within the same ensemble communicate. However, a baseline chip that uses multiple Vdd domains would also have the same issue: communicating clusters would have to cross f-domain boundaries.

One degree of freedom in Polyomino's assignment algorithm is whether the clusters that form the ensemble assigned to a job have to be physically contiguous. Not worrying about the physical layout simplifies the algorithm, but using non-contiguous clusters lengthens inter-thread communication. In our design, we consider two algorithms that give either high or low priority to choosing contiguous clusters for an ensemble. In practice, close-by clusters have a similar value of the systematic component of process variation. As a result, both algorithms try to pick contiguous clusters implicitly or explicitly. We consider the algorithms in the next section.

4.3.2 Core Assignment Algorithm

We call our core assignment algorithm *P_Assign*. When a new job arrives, *P_Assign* assigns an ensemble of clusters to it at a single f and, typically, does not revisit the assignment during the job's lifetime. In this discussion, *P_Assign* tries to maximize MIPS/watt; other related metrics can be used instead.

P_Assign uses information from both hardware and application. The hardware information includes each cluster's static power (P_{STA}) and maximum frequency supported (f_{MAX}) at Vdd_{NOM} at a reference temperature (T_0). This information is generated at manufacturing-testing time. Other

information includes the instantaneous load of the chip (which clusters are busy) to determine the headroom to the maximum power and T constraints. The application information includes the number of cores requested (equal to the number of threads) and an estimate of the average IPC of these threads. This IPC only needs to be roughly approximate. It can be obtained from previous runs of the application or from the current run, and does not include long-duration spinning. The output of *P_Assign* is the chosen ensemble of clusters to run the job, plus the f at which these clusters should run — equal to the minimum of the f_{MAX} of the chosen clusters. The general approach is depicted in Figure 4.1(c).

To see how *P_Assign* works, assume that a job requests n cores. *P_Assign* must return an ensemble E of size $|E| = \lceil n/ClSize \rceil$ clusters, where *ClSize* depicts the cluster size. Naively, *P_Assign* could simply check all the possible groups of $|E|$ free clusters, and pick the group that delivers the maximum MIPS/watt at Vdd_{NOM}. *P_Assign* indeed relies on such exhaustive search, however, conducts it in an intelligent way, such that the search space gets pruned and the runtime complexity reduces significantly: Specifically, *P_Assign* would repeatedly pick one free cluster i (which can cycle at most at f_{MAX_i}), and combine it with the best selection of $|E-1|$ clusters among those that can cycle *faster* than i, to arrive at the ensemble E which maximizes MIPS/watt:

$$\begin{aligned} & max_E(\frac{MIPS}{watt}) \equiv \\ & \equiv min_E(\frac{watt}{MIPS}) \equiv min_E(\frac{\sum_E P_{STA} + \sum_E P_{dyn}}{IPC \times |E| \times ClSize \times f_{MAX\_i}}) \\ & \equiv min_E(\frac{\sum_E P_{STA} + C \times V_{ddNOM}^2 \times |E| \times f_{MAX\_i}}{IPC \times |E| \times ClSize \times f_{MAX\_i}}) \end{aligned} \tag{4.1}$$

At the time cluster i is considered, all variables of this formula are known except $\sum_E P_{STA}$, the total P_{STA} of the ensemble E to be formed. We know the f of E, f_{MAX_i}, as set by the slowest cluster, namely cluster i; the operating Vdd is fixed chip-wide to Vdd_{NOM}; the number of cores requested determines ensemble size $|E|$; an estimate of $IPC(f_{MAX\_i})$ is already available; and finally, C, average cluster capacitance, is proportional to the area, and does not depend on the selection. $\sum_E P_{STA}$, on the other hand, changes with the selection of the clusters to form E. Thus, for each cluster i considered, the ensemble that maximizes MIPS/watt, $max_E(MIPS/watt)$, reduces to the ensemble of the clusters that deliver $min(\sum_E P_{STA})$. The pseudo-code for *P_Assign* is given in Algorithm 1: E^* is the optimal ensemble of $|E|$ clusters, while E is one selected candidate ensemble of $|E|$ clusters.

P_Assign can be implemented inexpensively, if the clusters are ordered offline from lowest to

Algorithm 1 Pseudo-code for *P_Assign*.

```
1: E* ← ∅
2: for each free cluster i in the chip do
3:    /* cluster i cycles at f_MAX_i */
4:    find the other |E| − 1 free clusters faster than i that have the minimum Σ_{E−1} P_STA
5:    /* these clusters can all cycle at f_MAX_i or higher */
6:    E ← { i ∪ these |E|-1 clusters }
7:    if (MIPS/watt(E) > MIPS/watt(E*)) then
8:       E* ← E
9:    end if
10: end for
```

highest P_{STA}, and from highest to lowest f_{MAX}. As *P_Assign* picks one cluster i at a time, it only needs to select, among those with higher f_{MAX}, the $|E| - 1$ ones that have the lowest P_{STA}. It then computes the MIPS/watt of the ensemble. This process is repeated once for each available cluster i, and the ensemble with the highest MIPS/watt is picked.

We design two *P_Assign* algorithms that differ in the priority given to choosing contiguous clusters for an ensemble. *P_Assign_NC* (for non-contiguous) is the algorithm just described, which gives low priority to selecting contiguous clusters; *P_Assign_C* (for contiguous) gives high priority to picking contiguous clusters. It does so by picking the ensemble of the clusters that deliver $min(\sum_E D_i \times P_{STA})$ as opposed to $min(\sum_E P_{STA})$. In this formula, D_i is the wavefront distance between a given component cluster of E and cluster i (Figure 4.1(d)). Consequently, clusters that are far apart are penalized and avoided. In this case, a practical implementation would order clusters not based on P_{STA}, but on $D_i \times P_{STA}$.

Both *P_Assign_NC* and *P_Assign_C* have a low overhead (Section 4.6.3 quantifies the number of instructions) and scale with $O(N^2)$, where N is the number of clusters per chip. Hence, they are fully practical.

For reference, we will compare them to two greedy algorithms: The non-contiguous version, *Greedy_NC*, picks free clusters in increasing order of P_{STA}/f_{MAX}. The contiguous version, *Greedy_C*, on the other hand, first assigns the free cluster of minimum P_{STA}/f_{MAX} as the center, and expands the ensemble along wavefronts of progressively increasing distance from the center. At each wavefront, *Greedy_C* picks clusters in increasing order of P_{STA}/f_{MAX}. These algorithms scale with O(N).

In contrast, a baseline manycore with per-cluster Vdd and f domains needs more complicated assignment algorithms. Specifically, assume that a job needs an ensemble of $|E|$ clusters. We need to find the set of clusters for the ensemble, and the ensemble's Vdd and f that maximize

MIPS/watt, assuming a single Vdd and f domain per ensemble to reduce complexity. To do so, we repeatedly pick one cluster i (which needs at least Vdd_{MIN_i} and can only cycle at f_{MAX_i} at Vdd_{MIN_i}). Then, we try to combine it with all of the possible groups of $|E|-1$ clusters among those that have a Vdd_{MIN} lower than Vdd_{MIN_i}. Recall that no cluster can operate safely below its designated Vdd_{MIN}. For each of these combinations, we set the ensemble's Vdd to Vdd_{MIN_i} (which is the highest one), raise the f of each of the $|E|-1$ clusters accordingly, and set the ensemble's f to the minimum of the resulting f of all of the $|E|$ clusters. We then compute the MIPS/watt. As we proceed, we pick the best selection for this i, and then the best selection over all possible i. We call this algorithm *MultipleVf_NC*. It has a higher overhead than *P_Assign* and scales with $O(N^3)$.

4.4 The Challenge of Applying Fine-Grain DVFS

A second challenge of not using multiple Vdd domains is that, without on-chip voltage regulators, the speed at which a DVFS algorithm can change Vdd is lower. Specifically, according to Kim *et al.* [77], on-chip regulators can change Vdd at a rate of about 30mV/ns. On the other hand, off-chip regulators take over two orders of magnitude longer to change the same Vdd magnitude. For example, a very conservative estimate is given by Intel's guidelines, which assume that off-chip regulators take 1.25μs to change Vdd by 6.25mV [78]. The reason for the lower speed is the higher latency of the communication between the CPU and the off-chip Vdd regulator. A Vdd regulator must sense the Vdd applied to cores to adjust its output Vdd accordingly; however, sensing the Vdd from off-chip is much slower than from on-chip. This limits the speed of Vdd regulators and thus requires a bulkier inductor and capacitor to support high efficiency. Overall, this inability to change Vdd fast could result in less energy-efficient DVFS operation under Polyomino than under multiple Vdd domains.

In practice, however, this issue may not have a significant effect on the execution's energy efficiency. One reason is that the Vdd changes needed at NTC are likely to be small most of the time. This is because even modest Vdd changes quickly bring the execution to regimes that are either too energy-inefficient or too slow. Moreover, the value of Vdd at NTC likely needs to be capped for reliability reasons. In addition, the algorithm for selecting the DVFS levels in an environment with multiple Vdd domains will be more complicated and have non-negligible overhead. Finally, the speed of DVFS changes is limited by the PLL re-locking time for f change, which is at least

10μs [79]. Very fast Vdd regulators do not eliminate this critical path. All of these effects may hide the higher latencies of off-chip regulators. Section 4.6.4 quantifies the tradeoffs.

4.5 Evaluation Setup

We evaluate Polyomino by modeling an 11nm NTC chip with 288 cores. The chip is organized in clusters. A cluster has 8 cores (each with a per-core private memory) and a cluster memory. The technology parameters are derived from ITRS and from projected trends from industry. Table 4.2 shows the technology and architecture parameters. The nominal values of Vdd and f are 0.55 V and 1.0 GHz (which would correspond to about 0.77 V and 3.3 GHz for STC). To model variation, we use the VARIUS-NTV [80] model, using $(\sigma/\mu)_{Vth} = 15\%$, $(\sigma/\mu)_{Leff} = 7.5\%$, and $\phi = 0.1$. Every single experiment is repeated for 100 chips with the same variation parameters but different variation profiles, and we present the average. More samples beyond 100 do not change the results noticeably.

System Parameters	
Technology node: 11nm Num. cores: 288 Num. clusters: 36 (8 cores/clus)	P_{MAX} = 100W T_{MAX} = 80°C Chip area ≈ 20mm x 20mm
Variation Parameters	
Correlation range: $\phi = 0.1$ Total $(\sigma/\mu)_{Vth} = 15\%$ Equal contrib. syst. & rand.	Sample size: 100 chips Total $(\sigma/\mu)_{Leff} = 7.5\%$ Equal contrib. syst. & rand.
Technology Parameters	
Vdd_{NOM} = 0.55V Vth_{NOM} = 0.33V f_{NOM} = 1.0GHz f network = 0.8GHz	On-chip Vdd regulator: 15% P loss Vdd guardband for V noise: 5% base + 5% if multiple Vdd domains
Architectural Parameters	
Core-private mem: 64KB WT, 4-way, 2ns access, 64B line Network: bus inside cluster and 2D-torus across clusters Cross a f domain boundary: 2ns Num. memory controllers: 8	Cluster mem: 2MB WB, 16-way, 10ns access, 64B line Coherence: directory-based MESI Avg. mem round-trip access time (before contention): ≈80ns

Table 4.2: Technology and architecture parameters.

Each core is a single-issue engine where memory accesses can be overlapped with each other and with computation. The per-core memories are private L1 caches, while the cluster memories

are shared L2 caches. We use a full-mapped directory-based MESI coherence protocol where each pointer corresponds to one cluster. The on-chip network is a bus inside a cluster and a 2D-torus across clusters. The chip has eight memory controllers.

To evaluate performance and power consumption, we interface Pin over a user-level pthreads library to the SESC [33] cycle-level architectural simulator. The power analysis relies on McPAT [61] scaled to 11nm. HotSpot is used to model the temperature. The algorithms for core assignment in the manycore are implemented in R [62].

For our experiments, we run multi-programmed workloads that contain the following PARSEC applications: blackscholes, ferret, fluidanimate, raytrace, swaptions, canneal, dedup, and streamcluster. Each application runs in parallel with a thread count that can range from 8 to 64 threads. We measure the complete parallel sections of the applications (i.e., region of interest) running the standard simsmall input data set. We report the average performance (e.g., in MIPS) or average energy efficiency (e.g., in MIPS/watt).

4.6 Evaluation

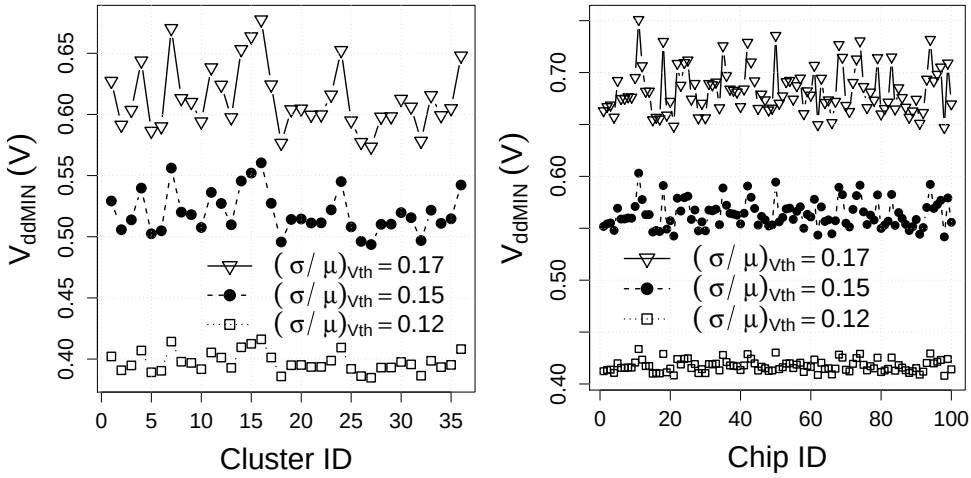
In this section, we first show the variation observed in NTC Polyomino chips, and then examine the effect of not having multiple Vdd domains, the impact of the core allocation algorithms, and the implications on fine-grain DVFS. Finally, we perform a sensitivity analysis of the architecture.

4.6.1 Variation Observed in NTC Polyomino Chips

To understand the results in the rest of the evaluation, this section assesses variation in Vdd_{MIN}, frequency (f), and static power (P_{STA}) in Polyomino chips. Each experiment covers three different values of process variation, $(\sigma/\mu)_{Vth}$ = 12, 15, and 17%, at two different scopes: (i) across the clusters within a representative Polyomino chip, namely, the chip with the median *chip-wide maximum* Vdd_{MIN} among the 100 chips analyzed; (ii) across 100 Polyomino chips.

Figure 4.2(a) depicts variation in cluster-wide maximum Vdd_{MIN} across the 36 clusters of a representative Polyomino chip. Higher values of $(\sigma/\mu)_{Vth}$ render not only a higher Vdd_{MIN} per cluster, but also a higher variation in Vdd_{MIN} across clusters (Figure 4.2(a)). Figure 4.2(b) characterizes variation in the chip-wide maximum Vdd_{MIN} across the 100 Polyomino chips analyzed. Both, maximum chip-wide Vdd_{MIN} and variation in maximum Vdd_{MIN} across chips, increase

with increasing $(\sigma/\mu)_{Vth}$ (Figure 4.2(b)).



(a) Vdd_{MIN} variation within chip

(b) Vdd_{MIN} variation across chips

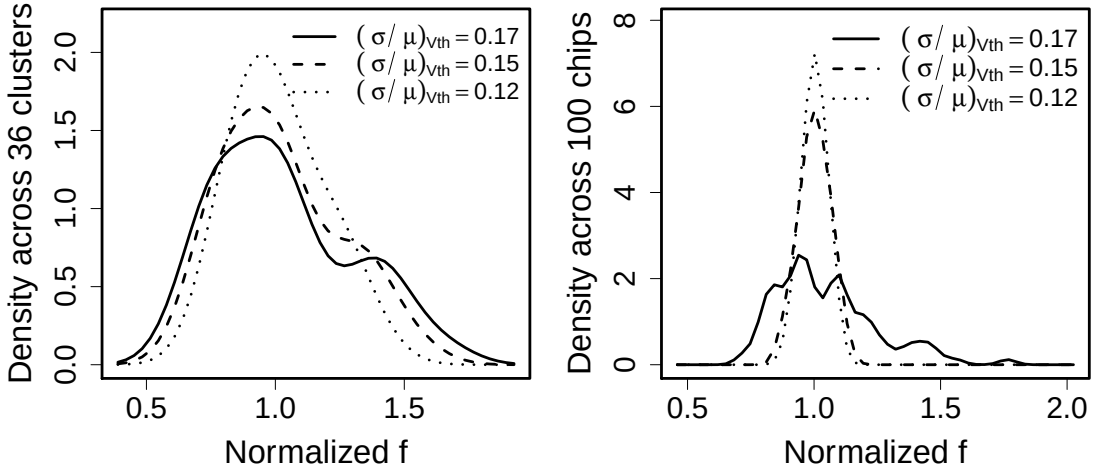
Figure 4.2: Variation of Vdd_{MIN} within a representative Polyomino chip (a), across 100 chips analyzed (b).

Maximum Vdd_{MIN} at different scopes (i.e cluster-wide or chip-wide) evolves as a function of $(\sigma/\mu)_{Vth}$, and determines the safe operating Vdd(s) per chip. Variation in f and P_{STA} indeed tightly depend on the operating Vdd(s). To demonstrate the impact of $(\sigma/\mu)_{Vth}$ on f and P_{STA} variation under a given *fixed* operating Vdd, Figures 4.3 and 4.4 deploy the chip-wide maximum Vdd_{MIN} for $(\sigma/\mu)_{Vth} = 0.17$ as the safe, fixed operating Vdd per chip across all configurations. We show kernel density estimates for each parameter [1]. In each plot, the x axis characterizes the normalized f or P_{STA} by the non-variation afflicted counterpart.

Figure 4.3(a) shows variation in cluster f, as determined by the slowest unit within each cluster, across the 36 clusters within a representative Polyomino chip. Across-cluster variation in cluster P_{STA} is given in Figure 4.4(a). On the other hand, Figure 4.3(b) depicts variation in chip-wide median cluster f; Figure 4.4(b), in chip P_{STA}, across 100 Polyomino chips. We observe that, as $(\sigma/\mu)_{Vth}$ increases, the spread of the distributions, hence the variation in f or P_{STA} increases.

Let us next explore how f and P_{STA} vary at different scopes considering the actual operating Vdd for different $(\sigma/\mu)_{Vth}$, as opposed to a fixed Vdd across all configurations. Figure 4.5(a) shows variation in cluster f; Figure 4.6(a), in cluster P_{STA}, across the 36 clusters within a representative Polyomino chip. On the other hand, Figure 4.5(b) depicts variation in chip-wide median cluster f;

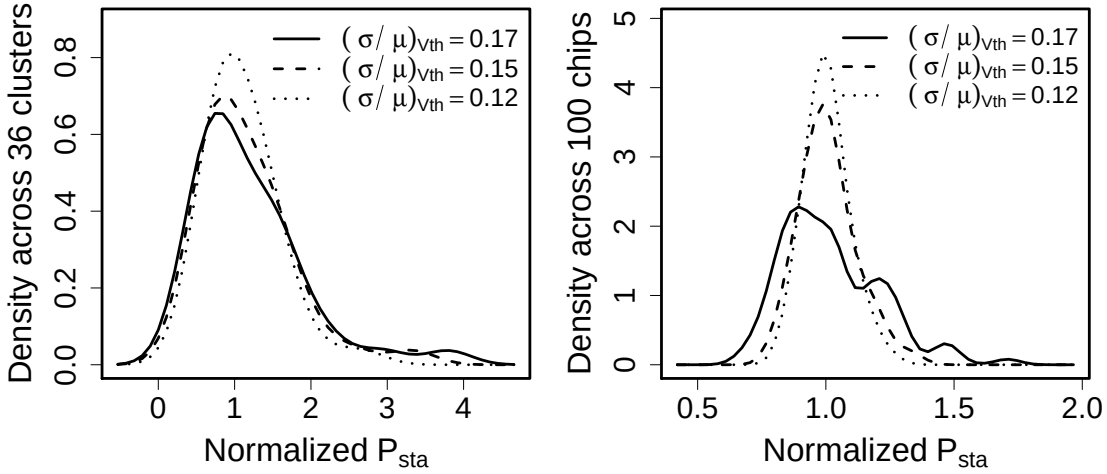
[1] Kernel density estimates can be regarded as smooth correspondents of histograms. While the area under each curve should be equal to 1, the density function can assume values > 1.



(a) f variation within chip

(b) f variation across chips

Figure 4.3: Kernel density of f within a representative Polyomino chip (a), across 100 chips analyzed (b).

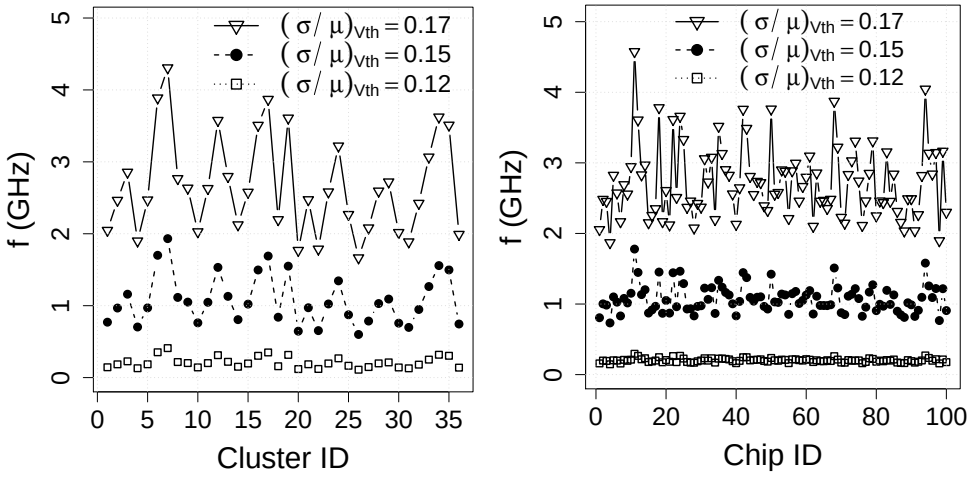


(a) P_{STA} variation within chip

(b) P_{STA} variation across chips

Figure 4.4: Kernel density of P_{STA} within a representative Polyomino chip (a), across 100 chips analyzed (b).

Figure 4.6(b), in chip P_{STA}, across 100 Polyomino chips.



(a) f variation within chip

(b) f variation across chips

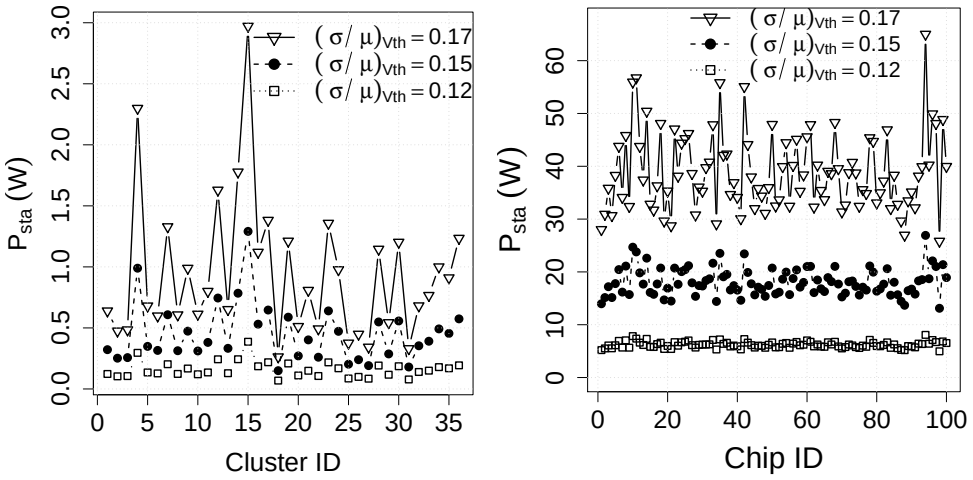
Figure 4.5: Variation of f within a representative chip (a), across 100 chips analyzed (b).

For each chip, all f and P_{STA} are evaluated at the chip-wide maximum Vdd_{MIN}; for the representative chip, V_{ddMIN} =416.1mV, 560.4mV, and 677.5mV, for $(\sigma/\mu)_{Vth}$ = 12, 15, and 17%, respectively. Chip-wide maximum Vdd_{MIN} represents the safe operating Vdd across all clusters. Recall that no Polyomino chip can operate at the high Vdd and fs imposed by $(\sigma/\mu)_{Vth}$ = 17% without exceeding the power budget. Due to the higher values of chip-wide maximum Vdd_{MIN}, progressively higher values of $(\sigma/\mu)_{Vth}$ not only induce higher cluster f and cluster P_{STA}, but also, increase variation in cluster f and cluster P_{STA} across clusters (Figure 4.5(a), Figure 4.6(a)). Similarly, with increasing $(\sigma/\mu)_{Vth}$, both, chip-wide median cluster f and chip P_{STA}, and variation across chips in chip-wide median cluster f and chip P_{STA}, increase (Figure 4.5(b), Figure 4.6(b)).

The rest of the paper uses the default $(\sigma/\mu)_{Vth}$ = 15% and the representative chip.

4.6.2 Effect of Not Having Multiple Vdd Domains

To see the effect of not supporting multiple Vdd domains in Polyomino, we compare the NTC environments of Table 4.3. *Perf* is a perfect environment with a Vdd and an f domain per cluster, without Vdd overheads. *Eff* is *Perf* plus the power inefficiencies of the on-chip Vdd regulators. By default, we use a 15% power loss in the regulators, which is in the ball park of the designs analyzed by Kim *et al.* [73] having a peak power efficiency of 77%. Recall that power loss of on-chip regulation changes as a function of the output Vdd. However, throughout the experiments, we



(a) P_{STA} variation within chip (b) P_{STA} variation within chip

Figure 4.6: Variation of P_{STA} within a representative chip (a), across 100 chips analyzed (b).

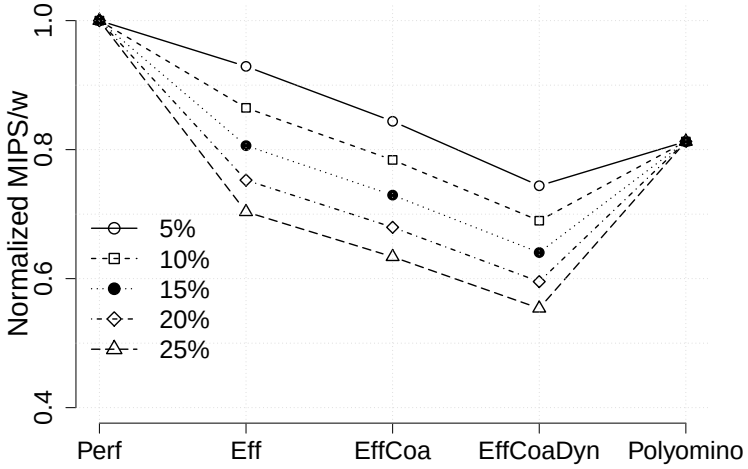
deploy the peak efficiency independent of the output Vdd to favor environments with multiple Vdd domains. *EffCoa* is *Eff* with coarser Vdd domains (four clusters per domain) due to practical implementation issues. *EffCoaDyn* is *EffCoa* plus a larger Vdd guardband. This is to tolerate potentially deeper dynamic Vdd droops — resulting from the likely lower capacitance of individual domains, compared to a large chip with a single Vdd domain. Extrapolating from James *et al.* [74], we add an additional 5% guardband over the base 5% Vdd-noise guardband used for the chip with a single Vdd domain [77]. Finally, Polyomino has a single Vdd domain. Note that all of the environments have one off-chip Vdd regulator.

Environment	Multiple Vdd Domains?	Description
Perf	Yes	Perfect environment with per cluster Vdd and f domains. No Vdd overheads.
Eff	Yes	*Perf* + power loss due to Vdd regulation.
EffCoa	Yes	*Eff* with coarse Vdd domains. Each one includes four clusters.
EffCoaDyn	Yes	*EffCoa* + a larger Vdd guardband to handle deeper dynamic Vdd droops.
Polyomino	No	Single Vdd domain. Each cluster is an f domain.

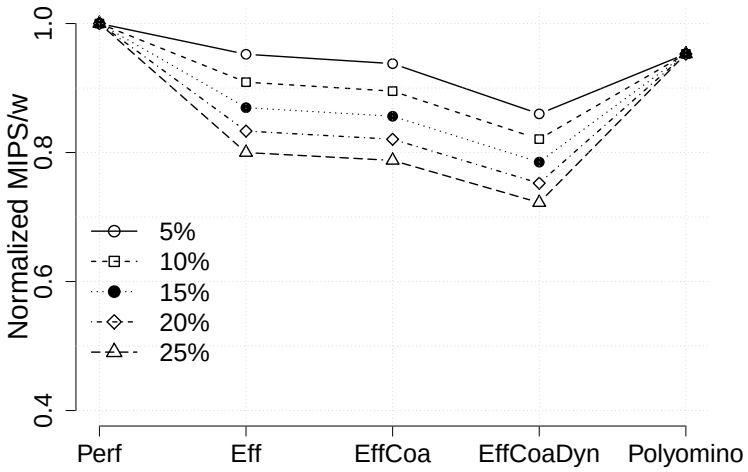
Table 4.3: Environments analyzed to assess Vdd domains at NTC.

Figure 4.7 compares the MIPS/w of our 288-core NTC chip for the different environments. We consider two scenarios: one where we use all 36 clusters (Figure 4.7(a)) and one where we use only 18 (Figure 4.7(b)). The workload consists of combinations of our PARSEC applications, where each

PARSEC code runs with 8 threads on one cluster. We report the geometric mean of MIPS/w across clusters. In each plot, we perform a sensitivity analysis of different power inefficiencies for the Vdd regulators (5%, 10%, 15%, 20%, and 25%, where 15% is the default). To make the comparison fair, the total power consumed by the chip and Vdd regulators in all of the environments and efficiency points is kept constant.



(a) Full utilization: All (36) clusters



(b) 50% utilization: 18 clusters

Figure 4.7: Normalized MIPS/w in the different environments for workloads that use all 36 clusters (a) or only 18 (b). We consider different Vdd regulator inefficiencies.

For each environment, we compute the sustainable per-domain Vdd and f as per Section 4.2.2: SRAM hold and write stability determine Vdd, and then timing tests in the SRAM and logic determine f. In this section, we are after the maximum MIPS/w each environment can achieve.

To exclude sub-optimal operation due to algorithmic imperfections, we deploy oracular core assignment algorithms tailored for each environment. The combinatorial optimization algorithms are based on the Hungarian method [81]. We ignore all algorithmic overheads. Implicitly, this approach favors the non-Polyomino environments, which require more complex core assignment algorithms due to the larger number of degrees of freedom (as induced by multiple voltage values per chip). Finally, after we compute the MIPS/w of each environment, we plot it normalized to that of *Perf* for the given utilization profile (100% or 50%).

Consider first the fully-utilized chip (Figure 4.7(a)). Polyomino only attains about 81% of the MIPS/w of *Perf*. This is because it does not exploit the multiple Vdd domains of *Perf* and, therefore, operates less energy-efficiently. However, in the environments with multiple Vdd domains, as we go from *Perf* to *Eff*, *EffCoa* and *EffCoaDyn*, progressively adding more realistic assumptions, the MIPS/w keep decreasing. By the time we reach a realistic environment (*EffCoaDyn*), its MIPS/w at 15% regulator power loss is 64% of *Perf*'s — significantly lower than Polyomino's.

As we vary the regulator power losses between 5% and 25%, we see that Polyomino always has a higher MIPS/w than the realistic *EffCoaDyn*. For the multiple Vdd domain chip to beat Polyomino, it must not require any increase in Vdd guardband (*EffCoa*) and use regulators with only 5% power losses. Alternatively, it must support per-cluster Vdd domains and no increase in Vdd guardband (*Eff*) and use regulators with at most 10% power losses.

Figure 4.7(b) repeats the experiment when we only need to use half of the clusters. We see similar trends for all the non-*Perf* environments except that the drop in MIPS/w is not as large. The reason is that each environment now picks a subset of energy-efficient clusters — leaving energy-inefficient ones idle. Polyomino attains 95% of the MIPS/w of *Perf*, while *EffCoaDyn* only attains 79% at 15% regulator power loss.

4.6.3 Core Assignment in Polyomino

The previous section showed that Polyomino delivers higher MIPS/w than a realistic implementation of multiple Vdd domains. In this section, we explore how to best use a Polyomino environment, by evaluating the different Polyomino core assignment algorithms of Section 4.3.2: *P_Assign_NC*, *P_Assign_C*, *Greedy_NC*, and *Greedy_C*. As a reference, we also show *MultipleVf_NC* applied to *EffCoaDyn* for three different Vdd regulator inefficiencies (5%, 10%, and 15%). In the previous section, the workload contained mixes of 8- threaded programs only, to expose the im-

pact of per-cluster Vdd and f domains. Now, we consider a flexible workload composed of a mix of applications from PARSEC, such that one runs with 64 threads, one with 32, one with 16, and one with 8. The mix uses a total of 15 clusters, and we measure MIPS/w. We run many experiments, forming similar mixes with all of the possible permutations of the PARSEC programs, and report the average MIPS/w. The power budget is set to the worst-case power consumed by all 36 clusters.

Our Polyomino algorithms are light-weight. For these experiments, *P_Assign_NC*, *P_Assign_C*, *Greedy_NC*, and *Greedy_C* execute, on average, 7835, 7912, 233, and 326 assembly instructions per run, respectively. The corresponding number for *MultipleVf_NC* is 153100.

Figure 4.8 shows the MIPS/w attained by the algorithms for three different manycore utilization scenarios: In the first one, all the clusters are available at the time the workload is launched (0% busy clusters). Hence, the whole power budget is available. The second scenario assumes that 25% of the clusters are already busy, and the third one that 50% are, running another, existing load when we launched our workload. Such load was 8-threaded runs of our highest-IPC application, namely dedup. For the 25% and 50% scenarios, the figure augments the bars with ranges. The range top corresponds to when the busy clusters were the least MIPS/w-efficient ones; the range bottom when they were the most MIPS/w-efficient ones. This choice matters because it sets the power budget available, and our load runs on the remaining clusters. In the figure, all the bars are normalized to *P_Assign_NC* with 0% use.

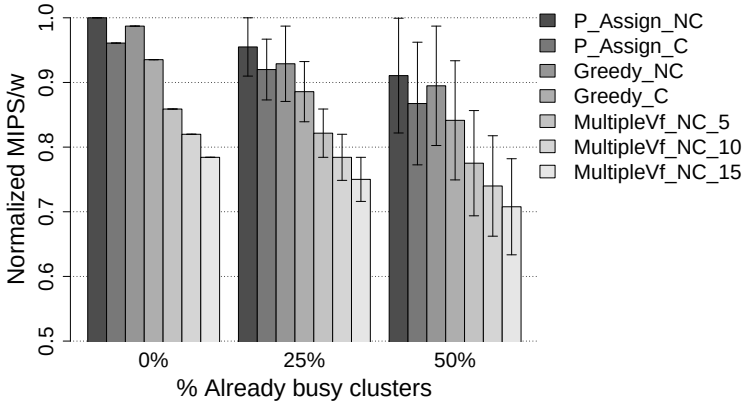


Figure 4.8: MIPS/w attained by different core assignment algorithms if 0%, 25%, 50% of the clusters were already busy initially. For the latter 2, the top (bottom) error bar depicts the MIPS/w if the least (most) energy-efficient clusters were initially busy.

Figure 4.8 shows that *P_Assign_NC* delivers the highest MIPS/w. *P_Assign_C* is about 5% worse.

The reason is that the advantages of picking close-by clusters are offset by the fact that some of these clusters are not the most energy-efficient ones available. On the other hand, energy efficiencies rendered by the greedy algorithms remain, on average, within 5% of the energy efficiencies of the corresponding *P_Assign* variants. This shows that while the latter's more accurate algorithms are worthwhile, lighter weight greedy algorithms can also be safely adapted.

The algorithm for the multiple Vdd domains (*MultipleVf_NC*) delivers a lower MIPS/w than *P_Assign* for 5%, 10% and, especially, 15% power inefficiency of the Vdd regulator. For the 15% inefficiency, *MultipleVf_NC* delivers around 78% of the MIPS/w of *P_Assign_NC*. Even though *MultipleVf_NC*'s overhead is still tolerable for our chip size, it suffers because the manycore's energy efficiency is low. Overall, *P_Assign_NC* (or *P_Assign_C*) is the most effective despite not supporting multiple Vdd domains.

As we increase the utilization of the chip, the impact of initial load increases, and the rate of MIPS/w delivered goes down. However, the relative trends across environments remain. One observation is that the MIPS/w delivered depends significantly on what clusters were busy with other work.

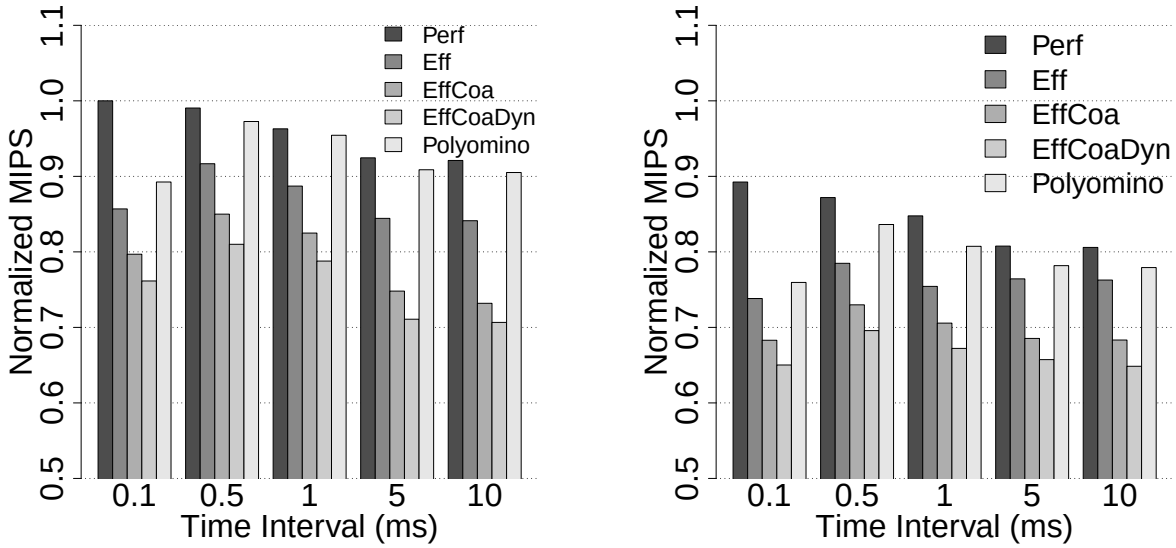
4.6.4 Implications on Fine-Grain DVFS

We next compare the application of fine-grain DVFS in the different environments of Table 4.3. We want to show the difference in the maximum MIPS that each environment can attain, hence we rely on perfect algorithms, ignoring all algorithmic overheads. We consider different intervals between DVFS adaptations (from 0.1ms to 10ms) and two classes of workloads (one that uses 16 clusters and one 24). A workload consists of a group of 8-threaded applications from PARSEC, where each application runs on a cluster. First, we identify the most MIPS/w-efficient (16 or 24) clusters, because these are expected to respond with the highest Δf increase to a given ΔVdd increase (and to the corresponding increase in power). Hence, these clusters are expected to deliver the maximum MIPS by application of DVFS. Next, the applications are statically assigned to the subset most MIPS/w-efficient clusters deploying the Hungarian algorithm [81]. The assignment is then fixed.

Since using a specific DVFS algorithm to dynamically tune Vdd and f can render sub-optimal operating points, we use a best-case Oracle algorithm which relies on non-linear optimization in finding fs and Vdds [82]. Moreover, we disregard any overhead involved in computing the

next interval's Vdd and f — which favors the more complex multi-Vdd environments with more degrees of freedom such as *EffCoaDyn*. Specifically, at the beginning of each interval, we assume we know the average IPCs of the applications in the interval. Then, we find the f and Vdd for each cluster (or a single Vdd in Polyomino) that delivers the highest overall MIPS — while remaining within the chip's power envelope and within the Vdd cap (aggressively set to 20% higher than the Vdd_{NOM} of Polyomino.)

Figure 4.9 shows the resulting performance of fine-grain DVFS for the different environments and adaptation intervals. Chart (a) and (b) correspond to loads that use 16 or 24 clusters, respectively. The performance for all the environments is normalized to *Perf* for 0.1ms and 16 clusters. As the DVFS adaptation interval decreases, higher MIPS would be expected due to more accurate tracking of application characteristics, were there no adaptation overhead. The impact of adaptation overhead on execution time, however, increases with decreasing DVFS interval. At progressively shorter intervals, as adaptation overhead becomes dominant, MIPS would start to deteriorate.



(a) 16 clusters

(b) 24 clusters

Figure 4.9: Performance of fine-grain DVFS under different environments.

The figures show that, while Polyomino's performance is lower than the perfect *Perf* environment, it is higher that the realistic *EffCoaDyn* environment. This is the case for all the adaptation intervals and the two chip utilization scenarios. For example, for 0.5–5ms intervals, Polyomino's performance is 20-28% higher than *EffCoaDyn* for 16 clusters. This means that the higher latency of

off-chip Vdd regulation in Polyomino is effectively hidden. There are two reasons for this. First, the magnitude of the Vdd changes needed is modest — in our case, they are capped at 20% of Vdd_{NOM}. Second, the overhead of PLL re-locking is in the critical path.

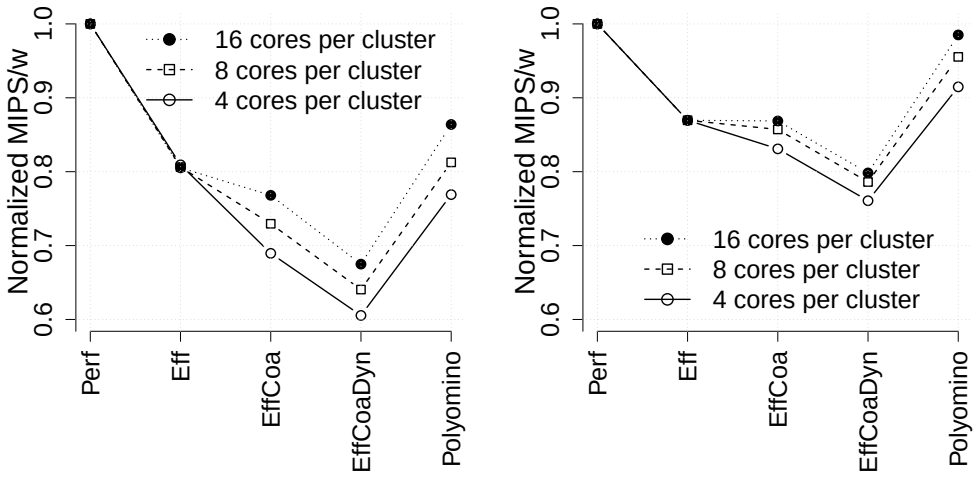
Only when the adaptation interval is very small (0.1ms), Polyomino's overhead is more visible and, therefore, its performance becomes comparable to *EffCoaDyn*'s. Finally, as we go from 16 to 24 clusters, the performance decreases across the board. The reason is that Vdd cannot go as high as for the 16-cluster case because the power headroom is lower due to the increased number of clusters. Still, Polyomino's performance is higher than *EffCoaDyn*'s.

4.6.5 Sensitivity Analysis

Finally, we assess the impact of cluster granularity (core count per cluster) for fixed core count per chip, and of core count per chip for fixed cluster granularity on energy efficiency. We deploy the default regulator power loss of 15% across experiments. The number of coarse grain Vdd domains per chip (for *EffCoa* and *EffCoaDyn*) is fixed to 9 not to increase complexity and Vdd guardbands any further.

First, we compare the energy efficiency of the environments in Table 4.3 for the 288-core Polyomino chip with either 4, 8 (default) or 16 cores per cluster. The results, shown in Figure 4.10, are organized as in Figure 4.7. The workload is the same as in Figure 4.7, except that the PARSEC applications run either with 4 threads or with 16 threads for 4 and 16 cores per cluster. All clusters are busy in Figure 4.10(a), and only the half in Figure 4.10(b). Each plot is normalized by the MIPS/w of *Perf* for the corresponding cluster granularity.

In each case Polyomino's energy-efficiency remains closer to *Perf* when compared to *EffCoaDyn*. As the cluster granularity decreases, the difference in MIPS/w of *Perf* and the rest of the environments reduces, since the difference between Vdd domains of different granularity becomes less pronounced under large core count per cluster. Specifically, under full utilization, MIPS/w of *EffCoaDyn* remains within 61%, 64%, and 68% of *Perf*, where Polyomino attains 77%, 81%, and 86% for 4, 8 and 16 cores per cluster. Under 50% utilization, the MIPS/w drop over *Perf* across different environments reduces, since each environment can now pick an energy-efficient subset of clusters, leaving energy-inefficient ones idle. For both of the utilization profiles, a similar MIPS/w drop over *Perf* applies for *Eff* independent of cluster granularity, because *Eff* differs from *Perf* only by the regulator power loss, not cluster organization.

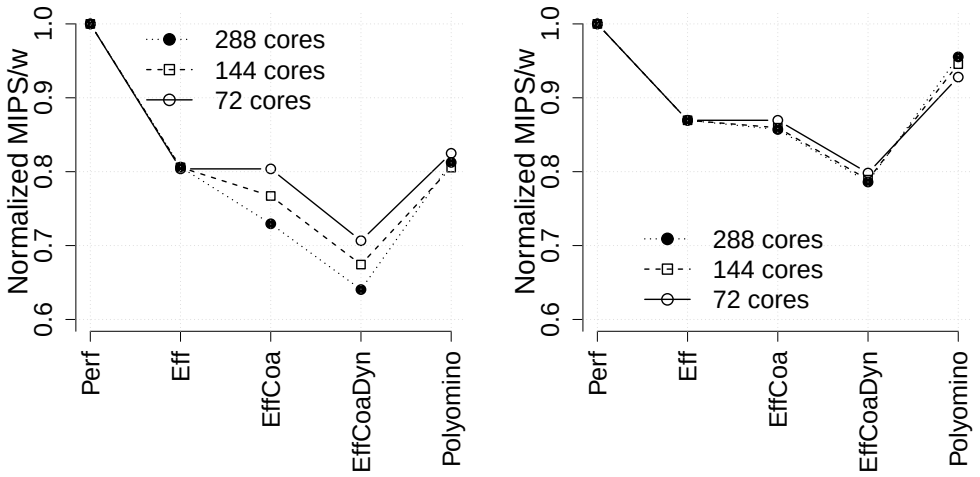


(a) Full utilization (b) 50% utilization

Figure 4.10: MIPS/w across different environments for a 288-core chip with 4, 8 (default) and 16 cores per cluster; under full utilization (a) and 50% utilization (b).

Second, we compare the energy efficiency of the environments in Table 4.3 for a 288-, 144- and 72-core Polyomino chip with 8 cores per cluster. The results, shown in Figure 4.11, are organized as in Figure 4.7. The workload is the same as in Figure 4.7. All clusters are busy in Figure 4.11(a), and only the half in Figure 4.11(b). Each plot is normalized by the MIPS/w of *Perf* for the corresponding core count.

In each case Polyomino's energy-efficiency remains closer to *Perf* when compared to *EffCoaDyn*. MIPS/w of *EffCoaDyn* over *Perf* decreases with increasing core count. This is because variation across chip increases with increasing core count. In addition, core count per coarse-grain domain increases as core count per chip increases, since we fix the number of coarse-grain domains to 9. On the other hand, for 72 cores per chip, each Vdd domain constitutes of one cluster only under *EffCoaDyn*. Even then, *EffCoaDyn* cannot beat Polyomino. Under full utilization, MIPS/w of *EffCoaDyn* remains within 70.7%, 67.4%, and 64.0% of *Perf*, where Polyomino attains 82.5%, 80.6%, and 81.3% for 72, 144, and 288 cores, respectively. Under 50% utilization, the MIPS/w drop over *Perf* across different environments reduces, since each environment can now pick an energy-efficient subset of clusters.



(a) Full utilization (b) 50% utilization

Figure 4.11: MIPS/w across different environments for 72-, 144- and 288-core chips with 8 cores per cluster; under full utilization (a) and 50% utilization (b).

4.7 Discussion

Our data supports that a large NTC manycore with a single Vdd domain can be more energy-efficient than one with many Vdd domains using on-chip Vdd regulators. This observation may change if new technologies come along that affect the fundamental tradeoffs involved.

One currently popular issue is that the upcoming use of FinFETs may reduce the amount of leakage power and process variations. However, both, share of leakage power and the impact of process variations will be still considerable in future technologies. For example, FinFETs can reduce the fraction of leakage power in total power only to 18% when we assume that the share of leakage power in STC processors with planar FETs is 30% in a 20nm technology [83]. Furthermore, the Vth sensitivity of FinFETs to L_{eff} systematic variations is still comparable to that of planar FETs, while the Vth sensitivity to W_{strip}, i.e., the width of each fin is far more significant than planar FETs, potentially leading to much higher leakage and delay variations [84]. Consequently, even with FinFETs, the impact of process variations on leakage and delay will continue to be important, and energy-efficient solutions to deal with them will be crucial.

4.8 Related Work

The most relevant proposals apply typically to non-NTC environments: Rotem *et al.* analyzed under power delivery limitations [71], and proposed a clustered topology to maximize performance, but ignored the impact of variation. Yan *et al.* suggested to deploy fast but power-inefficient on-chip Vdd regulation only for applications requesting fast DVFS, and to rely on slow but power-efficient off-chip regulation otherwise [85]. Although a much smaller chip than Polyomino was considered, the area overhead had to be justified by exploitation of area which would otherwise remain as dark silicon. In addition, the metal complexity is significantly higher than Polyomino. All of this body of work assumes an STC environment. On the other hand, Zhai *et al.* proposed a clustered CMP for NTC [35]. Since SRAM has a higher Vdd_{MIN} than logic, they let caches operate at a higher f than logic. This approach is costly, since (i) SRAM and logic are highly interleaved in the layout, (ii) extra voltage regulators are required; and not scalable into smaller technologies where the difference between SRAM and logic Vdd_{MIN} increases, diminishing the power savings at NTC. Jain *et al.* demonstrated an experimental IA chip capable to operate at NTC [86], but the chip has a single core only. Miller *et al.* deployed dual Vdd rails to handle variations at NTC [87]. Depending on variation profile and chip load, cores switch between two rails in a 32-core chip. The scalability into a larger number of cores may be questionable due to increased complexity.

4.9 Summary

We examined how to effectively cope with process variations in energy-efficient, future NTC chips. Our analysis suggests a novel, counter-intuitive approach to design such chips, without multiple Vdd domains, and with only f domains.

First, we showed that a chip with only f domains like Polyomino can deliver higher performance per watt than a chip with multiple Vdd and f domains supported by conventional on-chip Vdd regulators. The reasons are: (i) the regulators' power inefficiency, (ii) the increased Vdd guardband required by the finer grain Vdd domains to handle deeper dynamic Vdd droops (due to lower capacitance per Vdd domain), and (iii) the practical fact that a Vdd domain still includes many cores.

Second, we introduced core assignment algorithms for a Polyomino-type architecture that deliver high performance per watt while being simple. Finally, we showed that the higher latency of

Vdd changes during DVFS without on-chip *Vdd* regulators is effectively hidden. Key reasons are that the magnitude of the *Vdd* changes needed is modest and that PLL re-locking time is in the critical path.

5 CONCLUSION

To push back the many-core power wall, this thesis made the following contributions: First, it introduced *Dynamic Voltage Scaling for Aging Management* (DVSAM) — a new scheme for managing processor aging by tuning Vdd (but not the frequency), exploiting any currently-left aging guardband. The goal can be one of the following: consume the least power for the same performance and service life; attain the highest performance for the same service life and within power constraints; or attain even higher performance for a shorter service life and within power constraints.

Second, we presented *BubbleWrap*, a novel many-core architecture that makes extensive use of DVSAM to push back the many-core power wall. BubbleWrap selects the most power-efficient set of cores in the die — the largest set that can be simultaneously powered-on — and designates them as Throughput cores. They are dedicated to parallel-section execution. The rest of the cores are designated as Expendable. They are dedicated to accelerating sequential sections. BubbleWrap applies DVSAM in several environments. In some of them, it attains maximum sequential acceleration by sacrificing Expendable cores one at a time, running them at elevated Vdd for a significantly shorter service life each, until they completely wear-out.

In simulated 32-core chips, BubbleWrap provides substantial gains over a plain chip with the same power envelope. On average, our most aggressive design runs fully-sequential applications at a 16% higher frequency, and fully-parallel ones with a 30% higher throughput. We are now extending DVSAM to also include changes in processor frequency with time. This improvement should deliver better design points. We are also working on a model to accurately capture the deviations from the nominal behavior within a logical set of cores.

To help confront process variations at the architecture level at NTC, we then presented the first microarchitectural model of process variations for NTC. The model, called *VARIUS-NTV*, extends an existing variation model for STC. It models how variation affects the frequency attained and power consumed by cores and memories in an NTC manycore, and the timing and stability faults in SRAM cells at NTC. The key aspects include: (i) adopting a gate-delay model and an SRAM cell type that are tailored to NTC, (ii) modeling SRAM failure modes emerging at NTC, and (iii)

accounting for the impact of leakage in SRAM failure models.

We evaluated a simulated 11nm manycore at both NTC and STC. Our results showed that the expected process variations induce higher differences in f and power at NTC than at STC. For example, the maximum difference in tile f within a chip is ≈3.7x at NTC and only ≈2.3x at STC. We evaluated different core-tiling organizations in the chip and different configurations of on-chip Vdd- and f-domains. Our experiments showed that variation management is more crucial at NTC. Finally, we validated our model against an experimental 80-core prototype chip.

Next, we examined how to effectively cope with process variations in energy-efficient, future NTC chips. Our analysis suggests a novel, counter-intuitive approach to design such chips, without multiple Vdd domains, and with only f domains.

First, we showed that a chip with only f domains like Polyomino can deliver higher performance per watt than a chip with multiple Vdd and f domains supported by conventional on-chip Vdd regulators. The reasons are: (i) the regulators' power inefficiency, (ii) the increased Vdd guardband required by the finer grain Vdd domains to handle deeper dynamic Vdd droops (due to lower capacitance per Vdd domain), and (iii) the practical fact that a Vdd domain still includes many cores.

Second, we introduced core assignment algorithms for a Polyomino-type architecture that deliver high performance per watt while being simple. Finally, we showed that the higher latency of Vdd changes during DVFS without on-chip Vdd regulators is effectively hidden. Key reasons are that the magnitude of the Vdd changes needed is modest and that PLL re-locking time is in the critical path.

REFERENCES

[1] R. Dennard et al., "Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions," in *Journal of Solid-State Circuits*, October 1974.

[2] M. Horowitz et al., "Scaling, Power, and the Future of CMOS," in *International Electron Devices Meeting*, December 2005.

[3] E. J. Nowak, "Maintaining the Benefits of CMOS Scaling When Scaling Bogs Down," in *IBM Journal of Research and Development*, March/May 2002.

[4] International Technology Roadmap for Semiconductors, *2008 Update*.

[5] Intel Corporation, "Intel Core i7 Processor Extreme Edition and Intel Core i7 Processor Datasheet," November 2008.

[6] K. Bernstein et al., "High-Performance CMOS Variability in the 65-nm Regime and Beyond," in *IBM Journal of Research and Development*, July/September 2006.

[7] J. Abella, X. Vera, and A. González, "Penelope: The NBTI-Aware Processor," in *International Symposium on Microarchitecture*, December 2007.

[8] M. Agarwal et al., "Circuit Failure Prediction and Its Application to Transistor Aging," in *VLSI Test Symposium*, May 2007.

[9] J. Blome et al., "Self-Calibrating Online Wearout Detection," in *International Symposium on Microarchitecture*, December 2007.

[10] J. Shin et al., "A Proactive Wearout Recovery Approach for Exploiting Microarchitectural Redundancy to Extend Cache SRAM Lifetime," in *International Symposium on Computer Architecture*, June 2008.

[11] J. C. Smolens et al., "Detecting Emerging Wearout Faults," in *Workshop on Silicon Errors in Logic - System Effects*, 2007.

[12] J. Srinivasan et al., "The Case for Lifetime Reliability-Aware Microprocessors," in *International Symposium on Computer Architecture*, June 2004.

[13] J. Srinivasan et al., "Exploiting Structural Duplication for Lifetime Reliability Enhancement," in *International Symposium on Computer Architecture*, June 2005.

[14] A. Tiwari and J. Torrellas, "Facelift: Hiding and Slowing Down Aging in Multicores," in *International Symposium on Microarchitecture*, November 2008.

[15] F. Arnaud et al., "32nm General Purpose Bulk CMOS Technology for High Performance Applications at Low Voltage," in *Electron Devices Meeting*, December 2008.

[16] C. Auth, "45nm High-k + Metal Gate Strain-Enhanced CMOS Transistors," in *Custom Integrated Circuits Conference*, September 2008.

[17] W. Wang et al., "Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology," in *Transactions on Device and Materials Reliability*, December 2007.

[18] T. Sakurai and A. Newton, "Alpha-Power Law MOSFET Model and Its Applications to CMOS Inverter Delay and Other Formulas," in *Journal of Solid-State Circuits*, April 1990.

[19] D. Bergstrom, "45nm Transistor Reliability," in *International Technology Journal*, June 2008.

[20] S. Pae et al., "BTI Reliability of 45 nm High-k + Metal-Gate Process Technology," in *International Reliability Physics Symposium*, May 2008.

[21] J. W. McPherson, "Reliability Challenges for 45nm and Beyond," in *Design Automation Conference*, July 2006.

[22] E. Karl et al., "Compact In-Situ Sensors for Monitoring Negative-Bias-Temperature-Instability Effect and Oxide Degradation," in *International Solid-State Circuits Conference*, February 2008.

[23] J. Keane, D. Persaud, and C. H. Kim, "An All-in-one Silicon Odometer for Separately Monitoring HCI, BTI, and TDDB," in *Symposium on VLSI Circuits*, June 2009.

[24] T.-H. Kim, R. Persaud, and C. Kim, "Silicon Odometer: An On-Chip Reliability Monitor for Measuring Frequency Degradation of Digital Circuits," in *Journal of Solid-State Circuits*, April 2008.

[25] K. Stawiasz, K. Jenkins, and P.-F. Lu, "On-Chip Circuit for Monitoring Frequency Degradation Due to NBTI," in *International Reliability Physics Symposium*, May 2008.

[26] M. Popovich, A. Mezhiba, and E. Friedman, *Power Distribution Networks with On-Chip Decoupling Capacitors*. Springer, 2008, ch. 15.

[27] J. Dorsey et al., "An Integrated Quad-Core Opteron Processor," in *International Solid-State Circuits Conference*, February 2007.

[28] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-level Power Analysis and Optimizations," *SIGARCH Computer Architecture News*, 2000.

[29] W. Huang et al., "Many-Core Design from a Thermal Perspective," in *Design Automation Conference*, July 2008.

[30] W. Zhao and Y. Cao, "New Generation of Predictive Technology Model for Sub-45nm Design Exploration," in *International Symposium on Quality Electronic Design*, 2006.

[31] A. Khakifirooz and D. Antoniadis, "MOSFET Performance Scaling Part II: Future Directions," in *Transactions on Electron Devices*, June 2008.

[32] R. Teodorescu and J. Torrellas, "Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors," in *International Symposium on Computer Architecture*, June 2008.

[33] J. Renau et al., "SESC Simulator," January 2005, http://sesc.sourceforge.net.

[34] R. Gonzalez, B. Gordon, and M. Horowitz, "Supply and Threshold Voltage Scaling for Low Power CMOS," in *Journal of Solid-State Circuits*, August 1997.

[35] B. Zhai et al., "Energy Efficient Near-Threshold Chip Multi-Processing," in *International Symposium on Low power Electronics and Design*, 2007.

[36] K. Fan et al., "Bridging the Computation Gap between Programmable Processors and Hardwired Accelerators," in *International Symposium on High Performance Computer Architecture*, February 2009.

[37] L. Chang et al., "Practical Strategies for Power-Efficient Computing Technologies," *Proceedings of the IEEE*, February 2010.

[38] R. G. Dreslinski et al., "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, February 2010.

[39] D. Markovic et al., "Ultralow-Power Design in Near-Threshold Region," *Proceedings of the IEEE*, February 2010.

[40] S. Dighe et al., "Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor," *Journal of Solid-State Circuits*, January 2011.

[41] E. Humenay et al., "Impact of Process Variations on Multicore Performance Symmetry," in *Conference on Design, Automation and Test in Europe*, April 2007.

[42] X. Liang and D. Brooks, "Mitigating the impact of process variations on processor register files and execution units," in *International Symposium on Microarchitecture*, December 2006.

[43] D. Marculescu and E. Talpes, "Variability and energy awareness: a microarchitecture-level perspective," in *Design Automation Conference*, June 2005.

[44] B. F. Romanescu et al., "Quantifying the Impact of Process Variability on Microprocessor Behavior," in *Workshop on Architectural Reliability*, December 2006.

[45] S. Sarangi et al., "VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects," *Transactions on Semiconductor Manufacturing*, February 2008.

[46] S. Borkar et al., "Parameter Variations and Impact on Circuits and Microarchitecture," in *Design Automation Conference*, June 2003.

[47] A. Srivastava et al., *Statistical Analysis and Optimization for VLSI: Timing and Power*. Springer, 2005.

[48] M. Eisele et al., "The Impact of Intra-die Device Parameter Variations on Path Delays and on the Design for Yield of Low Voltage Digital Circuits," *IEEE Transactions on VLSI Systems*, December 1997.

[49] Y. Cheng and C. Hu, *MOSFET Modeling and Bsim3 User's Guide*. Kluwer Academic Publishers, 1999.

[50] D. Ernst et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," in *International Symposium on Microarchitecture*, December 2003.

[51] K. A. Bowman, B. L. Austin, J. C. Eble, X. Tang, and J. D. Meindl, "A Physical Alpha-power Law MOSFET Model," in *International Symposium on Low Power Electronics and Design*, 1999.

[52] Y. Cao and L. T. Clark, "Mapping Statistical Process Variations Toward Circuit Performance Variability: An Analytical Modeling Approach," in *Design Automation Conference*, 2005.

[53] H. Im, "Physical Insight Into Fractional Power Dependence of Saturation Current on Gate Voltage in Advanced Short Channel MOSFETS (Alpha-power Law Model)," in *International Symposium on Low Power Electronics and Design*, 2002.

[54] M. Orshansky, J. Chen, and C. Hu, "Direct Sampling Methodology for Statistical Analysis of Scaled CMOS Technologies," *Transactions on Semiconductor Manufacturing*, November 1999.

[55] C. C. Enz et al., "An Analytical MOS Transistor Model Valid in All Regions of Operation and Dedicated to Low-voltage and Low-current Applications," *Analog Integrated Circuits and Signal Processing*, July 1995.

[56] L. Chang et al., "An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches," *Journal of Solid-State Circuits*, April 2008.

[57] Y. Morita et al., "An Area-Conscious Low-Voltage-Oriented 8T-SRAM Design under DVS Environment," in *Symposium on VLSI Circuits*, 2007.

[58] S. Mukhopadhyay et al., "Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, December 2005.

[59] International Technology Roadmap for Semiconductors, *2009 Update*.

[60] C.-K. Luk et al., "Pin: Building Customized Program Analysis Tools With Dynamic Instrumentation," in *Conference on Programming Language Design and Implementation*, 2005.

[61] S. Li et al., "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *International Symposium on Microarchitecture*, December 2009.

[62] The R Project for Statistical Computing, *http://www.r-project.org/*.

[63] J. Abella et al., "High-Performance Low-Vcc In-Order Core," in *International Symposium on High Performance Computer Architecture*, January 2010.

[64] P. Friedberg et al., "Modeling Within-die Spatial Correlation Effects for Process-design Co-optimization," in *International Symposium on Quality of Electronic Design*, March 2005.

[65] Predictive Technology Model (PTM), *http://ptm.asu.edu/*.

[66] BSIM, *http://www-device.eecs.berkeley.edu/~bsim/BSIM4/BSIM460*.

[67] J. Donald and M. Martonosi, "Power Efficiency for Variation-tolerant Multicore Processors," in *International Symposium on Low Power Electronics and Design*, October 2006.

[68] S. Herbert and D. Marculescu, "Characterizing Chip-multiprocessor Variability-tolerance," in *Design Automation Conference*, June 2008.

[69] J. Li and J. Martinez, "Dynamic Power-performance Adaptation of Parallel Computation on Chip Multiprocessors," in *International Symposium on High-Performance Computer Architecture*, February 2006.

[70] K. Rangan et al., "Achieving Uniform Performance and Maximizing Throughput in the Presence of Heterogeneity," in *International Symposium on High Performance Computer Architecture*, February 2011.

[71] E. Rotem et al., "Multiple Clock and Voltage Domains for Chip Multiprocessors," in *International Symposium on Microarchitecture*, December 2009.

[72] H. A. Ghasemi et al., "Cost-Effective Power Delivery for Supporting Per-Core Voltage Domains for Power-Constrained Processors," in *Design Automation Conference*, June 2012.

[73] W. Kim et al., "A Fully-integrated 3-Level DC/DC Converter for Nanosecond-scale DVS with Fast Shunt Regulation," in *International Solid-State Circuits Conference*, February 2011.

[74] N. James et al., "Comparison of Split-Versus Connected-Core Supplies in the POWER6 Microprocessor," in *International Solid-State Circuits Conference*, February 2007.

[75] J. Lee and N. S. Kim, "Optimizing Total Power of Many-core Processors Considering Voltage Scaling Limit and Process Variations," in *International Symposium on Low Power Electronics and Design*, March 2009.

[76] R. R. Dobkin and R. Ginosar, "Two-phase Synchronization with Sub-cycle Latency," *Integration, the VLSI Journal*, June 2009.

[77] W. Kim et al., "System Level Analysis of Fast, Per-core DVFS Using On-chip Switching Regulators," in *International Symposium on High Performance Computer Architecture*, February 2008.

[78] *http://www.intel.com/Assets/en_US/PDF/designguide/321736.pdf*.

[79] A. Bashir et al., "Fast Lock Scheme for Phase-Locked Loops," in *Custom Integrated Circuits Conference*, September 2009.

[80] U. R. Karpuzcu et al., "VARIUS-NTV: A Microarchitectural Model to Capture the Increased Sensitivity of Manycores to Process Variations at Near-Threshold Voltages," in *International Conference on Dependable Systems and Networks*, June 2012.

[81] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[82] R. H. Byrd et al., "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal of Scientific Computing*, 1995.

[83] A. Keshavarzi et al., "Architecting Advanced Technologies for 14nm and Beyond with 3D FinFET Transistors for the Future SoC Applications," in *International Electron Devices Meeting*, December 2011.

[84] H. Khan et al., "Simulation of the impact of process variation on the optimized 10-nm FinFET," *Transactions Electron Devices*, August 2008.

[85] G. Yan et al., "AgileRegulator: A Hybrid Voltage Regulator Scheme Redeeming Dark Silicon for Power Efficiency in a Multicore Architecture," in *International Symposium on High Performance Computer Architecture*, February 2012.

[86] S. Jain et al., "A 280mV-to-1.2V Wide-Operating-Range IA-32 Processor in 32nm CMOS," in *International Solid-state Circuits Conference*, February 2012.

[87] T. Miller et al., "Booster: Reactive Core Acceleration for Mitigating the Effects of Process Variation and Application Imbalance in Low-voltage Chips," in *International Symposium on High Performance Computer Architecture*, February 2012.