Leveraging the Core-Level Complementary Effects of PVT Variations to Reduce Timing Emergencies in Multi-Core Processors

Guihai Yan
a) Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences (CAS), China
b) Graduate University of CAS
yan_guihai@ict.ac.cn

Xiaoyao Liang
NVIDIA Corporation
USA
xliang@nvidia.com

Yinhe Han, Xiaowei Li
a) Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences (CAS), China
b) Graduate University of CAS
{yinhes, lxw}@ict.ac.cn

ABSTRACT

Process, Voltage, and Temperature (PVT) variations can significantly degrade the performance benefits expected from next nanoscale technology. The primary circuit implication of the PVT variations is the resultant timing emergencies. In a multi-core processor running multiple programs, variations create spatial and temporal unbalance across the processing cores. Most prior schemes are dedicated to tolerating PVT variations individually for a single core, but ignore the opportunity of leveraging the complementary effects between variations and the intrinsic variation unbalance among individual cores. We find that the notorious delay impacts from different variations are not necessary aggregated. Cores with mild variations can share the violent workload from cores suffering large variations. If operated correctly, variations on different cores can help mitigating each other and result in a variation-mild environment. In this paper, we propose Timing Emergency Aware Thread Migration (TEA-TM), a delay sensor-based scheme to reduce system timing emergencies under PVT variations. Fourier transform and frequency domain analysis are conducted to provide the insights and the potential of the PVT co-optimization scheme. Experimental results show on average TEA-TM can help save up to 24% throughput loss, at the same time improve the system fairness by 85%.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; C.1.4 [**Processor Architectures**]: Parallel Architectures

General Terms

Reliability, Experimentation, Design

Keywords

Timing emergency, PVT variations, complimentary effects, delay sensor, thread migration

ISCA'10, June 19–23, 2010, Saint-Malo, France.

1. INTRODUCTION

CMOS technology scaling has been and will continue to be the main driving force in quest for performance in the computing industry by integrating smaller and faster transistors onto a single chip. The scaling trend, however, is greatly threatened by the ever-increasing parameter variations. Parameter variations induce delay violations in the system, which can eat up the performance gain from technology scaling [1]. Parameter variations can be classified into process, voltage, and temperature variations (PVT) [2]. Manufacturing and process imperfections cause process variation which makes each transistor slightly different in their delay and power profile. Voltage variation occurs when large current switch happens in the microprocessor that leads to supply voltage fluctuations through parasitic power delivery network. Temperature variation is mostly due to the imbalanced power consumption in a chip, which leads to different intra- or inter-core temperature. The primary circuit impact of PVT variations is the resultant delay violation or timing emergency (i.e. part of the microarchitecture circuit cannot meet the operating frequency). The traditional solution for the problem is to over-design the system based on the worst-case scenario. However, with the variations growing, the worst-case principle of design may cause too much design overhead and fabrication cost.

Recently, researchers have started to look for alternative solutions for PVT variations. Liang et al. proposed voltage interpolation [3] [4] and Teodorescu et al. proposed fine-grain body bias tuning at the microarchitecture level for mitigating the delay impact of process variation [5]. Powell et al. proposed pipeline damping [6] and Gupta et al. proposed delayed commit and rollback scheme for reducing or tolerating voltage variation [7]. Skadron et al. conducted some of the early work on processor temperature variation [8] and Donald et al. surveyed many different techniques for controlling the variation [9]. However, prior schemes are dedicated to tolerating PVT variations individually, but ignore the opportunity for leveraging the complementary effects between these variations. Since not all the variations affect circuit delay at the same time and to the same direction, we find that the delay impacts from different types of variations are not necessary aggregated. Moreover, the nature of parameter variations create intrinsic imbalance in variation tolerability across different cores in the processor. If operated correctly,

these variations can help compensate each other across cores and thereby result in a variation mild environment.

In this paper, we propose a new approach to mitigate the impact of PVT variations. Our approach, called Timing Emergency Aware Thread Migration (TEA-TM), aims to address the "real" timing emergencies which endanger the reliability of the processors. Our scheme is purely based on the circuit delay values measured by distributed delay sensors, waiving the need for any other sensing schemes such as temperature or voltages sensors. With the delay measured in individual cores, Fast Fourier Transform (FFT) is conducted and simple frequency analysis will provide the strength of each variation source (P, V, and T), deduced from the magnitude of corresponding frequency components on the frequency spectrum. The DC and low frequency components typically represent for process and temperature variations, while the high frequency components stand for the voltage variation. Optimizations are performed to smooth out the total variation strength across cores by exchanging their high frequency components through thread migration (TM). In time domain, this is equivalent to migrate voltage-violent threads to process- and temperature-mild cores which lead to overall reductions in timing emergencies. Since frequency analysis is hard to achieve in real-time, we discuss alternative algorithms with their hardware complexity and effectiveness. Overall, we make three contributions:

• Unlike the previous schemes targeting to PVT variations individually, TEA-TM seeks to leverage the spatial and temporal complementary effects between variations and across different cores. PVT variations have different space and time span that provides unique opportunity to cancel/mitigate their circuit impact — delay variation. Our results show migrating voltage-violent thread to process- and temperature-mild cores can greatly reduce the overall occurrence of timing emergencies.

• Unlike the previous schemes requiring different sensors for different variations, our scheme only relies on delay sensors, which provides the most faithful information for timing emergency. We can deduce the strength of different variations by performing simple frequency analysis. This solution is scaling friendly since we can apply the same method to new variation sources without deploying new types of sensors in the future.

• We present an analysis method from frequency domain perspective. Using this method, we can provide insights on how to leverage the complementary effects between variations and how to decide the optimal thread migration intervals. In addition, this paper shows that the frequency analysis can be a powerful tool for computer architects in dealing with variation-related issues.

The rest of the paper is organized as follows: Section 2 presents background information. Section 3 discusses the frequency and time domain perspective of the scheme. Section 4 presents the design challenges and the detailed design implementation. Section 5 presents the experiment setup, followed by simulation results in Section 6. Section 7 provides the related work and Section 8 concludes this paper.

2. BACKGROUND

2.1 PVT Variations

Process Variation. Chip manufacturing imperfections introduce process variation which makes transistors slightly differ in their delay characteristic. At system level, process variation leads to different maximum operating frequency for processing cores on a single die [10]. This phenomenon can also be interpreted as different cores have different tolerability to delay variation if they are configured at the same frequency. In other words, process-mild cores (i.e. faster cores) are able to tolerate more delay fluctuations. This unbalanced "tolerability" provides a new optimization opportunity especially for the future many-core architectures. Process variation is static and determined at the chip fabrication time. In a frequency domain analysis, it engenders DC component on the spectrum as explained in Section 3.

Voltage Variation. Voltage variation mainly results from program variability. Different application activity requires different amount of current. Variation in current demand transfers to voltage fluctuations through two physical mechanisms: IR-drop and Inductive Noise (a.k.a. $L\mathrm{d}i/\mathrm{d}t$ problem). The presence of parasitical capacitance and inductance makes a robust power delivery subsystem extremely difficult to implement. Voltage variation also affects circuit delay and causes delay unbalance among processing cores. Unlike the process and temperature variations, voltage variation is usually fast changing and represents high frequency components in frequency analysis.

Temperature Variation. The average and peak temperature of a processor core is highly application-specific [9]. Even for the same program, different phase of the application will generate different power consumption and temperature. In a multi-core processor, this creates temperature imbalance among cores which provides another optimization opportunity. Circuit delay is highly temperature-related [11]. Typically, a processing core can run faster with lower temperature. In other words, temperature-mild cores (i.e. cooler cores) are able to tolerate larger delay fluctuations. Temperature variation is usually slow time-varying and shows up to be low frequency components on the spectrum in frequency analysis.

The common impact of the three variations to multi-core processors is the delay variation observed among individual cores. But the negative delay impacts of the three variations are not necessary aggregated. Some application threads tend to be voltage-violent (note that "voltage violent" is not necessarily associated with higher power/temperature, and vice versa) and they cause large voltage fluctuations. If they happen to be running on slow or hot cores, the aggregated effect will make the cores very susceptible to timing errors. The chance of timing violations, however, can be greatly reduced if we migrate such threads to faster or cooler cores beforehand. Most of the prior works focus on measuring real-time temperature or voltage. Instead, we focus on measuring the real circuit delay. Since timing violation is the ultimate impact of temperature and voltage fluctuations, focusing directly on circuit delay brings the most reliable and confident design choice. In this paper, we will study the joint delay impact of the three variations, focus on leveraging the complimentary effect and propose a co-optimization scheme to reduce the timing emergencies.

2.2 Delay Sensors

One key concept of this paper is to infer the impact of PVT variations purely through circuit delay measurement, unlike the prior schemes [7][9] depending on slow temperature and voltage sensors. This brings three benefits to our scheme: 1) delay sensors naturally take the process variation into account; 2) delay sensors are much faster than thermal or voltage sensors, which is critical in triggering timely thread migration; 3) delay sensors save us the additional cost for adding temperature, voltage and other types of sensors.

Real-time delay values are provided by distributed delay

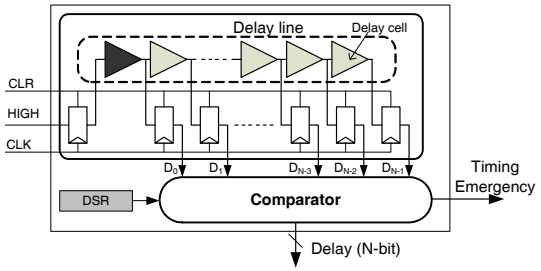


Figure 1: Conceptual delay sensor design

sensors. These sensors serve as canary circuits and the delay values measured represent critical circuit delays of the surrounding region. The measurements are highly reliable since the delay sensors share the same process corner, ambient temperature and voltage supply network with the critical paths in the same core. The key element of a delay sensor is Time-to-Digital Converter (TDC), as Figure 1 shows. TDC [12][13] is an appropriate and well-studied device for delay measurement. The measured resolution can easily reach 5ps with 90nm technology [13].

The basic working principle of delay sensors is describe as follows: At the effective edge of CLK, the signal 'HIGH' is triggered to propagate through the delay line. At the end of the cycle, the delay propagation is stamped by a series of flip-flops, represented as thermometer code: $D_0D_1 \cdots D_{N-1}$. Delay signature register (DSR) is used to store delay thresholds which are used for comparing against the sampled delay for emergency detection.

2.3 Thread Migration

Our scheme relies on thread migration (TM) which has been proved necessary for multi-core processors [14]. Not only can TM be engaged for thermal management [15], but also help steer the applications running in a more power-efficient way on multi-core processors [16]. Every migration involves transferring some states such as architectural registers, from one core to the other. The performance penalty imposed by migration largely depends on the target multi-core architectures. For light-weight cores with limited speculative capabilities, the migration penalty is much less than heavy-weight cores with aggressive speculation.

The performance penalty for heavy-weight cores can also be amortized well if the TM interval is kept above a threshold. Comprehensive case study [14] shows that for a multi-core processor with private L1 and shared L2 cache, TM interval of 2.5M instructions (0.825ms at 3GHz) or more makes the performance penalty negligible. Even with TM interval at 640K instructions (0.21ms at 3GHz), the worst-case overhead is no more than 15%. We will prove in this paper that this TM interval lines up well on the frequency spectrum between the slow varying process/temperature variations and the fast changing voltage variation. The optimum TM interval in our scheme only imposes marginal performance penalty. This serves as the basic rationale behind our TEA-TM scheme.

2.4 Rollback Recovery

Our scheme can reduce timing emergencies but cannot completely eliminate timing violations. Rollback recovery scheme [17][18] is applied when the circuit undergoes true timing errors. The architectural states of all active cores are check-pointed at the magnitude of tens of millisecond. Whenever an error is detected, the architecture states will be rolled back to the most recent check-point and re-run. This means

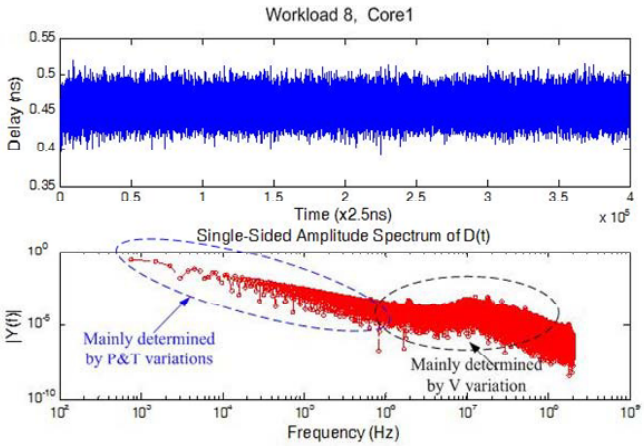


Figure 2: Frequency analysis of D(t)

an error can impose hundreds of millions cycles overhead under the worst-case. In this paper, we assume each core in the processor has individual checkpoint and rollback logic, just as previous proposed ReVive architecture [17].

3. FREQUENCY AND TIME DOMAIN PERSPECTIVE OF PVT VARIATIONS

In this section, we will discuss PVT variations at both frequency and time domain. Fourier Transform is performed on the sampled delay values and frequency analysis provides clear insight on how our proposed TEA-TM scheme can help to mitigate the delay variation.

3.1 Characterizing Timing Emergency under PVT Variations

At any time t, the critical delay $D(t)$ is determined by the real-time on-site variations. The variations include time-independent process variation P, slow-varying temperature variation $T(t)$ and fast-changing voltage variation $V(t)$, which is shown in Eq.(1).

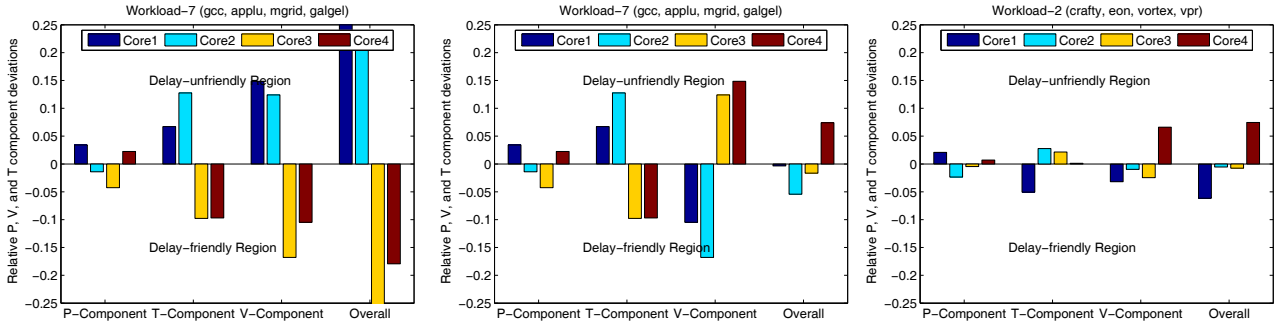
$$D(t) = f(P, V(t), T(t)) \tag{1}$$

We define the designed nominal voltage as V_{spec}, nominal temperature as T_{spec}, and the designed nominal delay as D_{spec}. At any time, the delay variation $\Delta D(t)$ is defined as the difference between the critical delay and the nominal delay $\Delta D(t) \triangleq D_t - D_{spec}$. It can be further decomposed into the impact of the three variations respectively using a linear model [11] shown in Eq.(2), where α, β, γ are experience constants.

$$\Delta D(t) = D(t) - D_{spec} = \alpha P + \beta(V(t) - V_{spec}) + \gamma(T(t) - T_{spec}) \tag{2}$$

We further defined timing emergency occurs when $\Delta D(t)$ is larger than a predetermined threshold D_{TH}. This happens if the delay variation becomes big enough and generates timing violations in the circuit. The Emergency Level (EL) is defined as the total number of timing emergencies per every 100 millions cycles. Higher EL means more delay violations which may lead to larger performance loss.

To conduct Discrete Fourier Transform (DFT), we sample $D(t)$ by delay sensors and obtain discrete delay values $D(n)$. Eq.(3) shows Fourier Transform, where $\omega = 2\pi \times frequency$. $Y(e^{j\omega})$ is called the frequency spectrum of $D(n)$. The strength of each frequency component is expressed with the $|Y(e^{j\omega})|$.



(a) Large potential, but without optimization (b) Large potential, with optimization (c) Little optimization potential

Figure 3: Relative frequency spectrum deviations of P, V, and T components in 1ms execution interval on a 2GHz quad-core processor. The boundary frequencies for P-Component: 0-100Hz, T-Component: 100Hz-1MHz, V-Component: 1MHz-250MHz.

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{\infty} D(n)e^{-j\omega n} \quad (3)$$

3.2 Frequency Domain Analysis

Timing variation is determined by voltage variation (V component), temperature (T component), and process variations (P component). From frequency domain, qualitatively, P component clearly represents the DC component, T component contributes to low-frequency components due to millisecond thermal constant of silicon material, while V component dominates the high-frequency components due to its much faster circuit switching activities.

Figure 2 shows the critical delay fluctuation in a microprocessor for 1ms. We conduct FFT on the delay values, and show the corresponding frequency spectrum in the below subfigure, where both the frequency and amplitude are plotted in logarithmic scale. The amplitude in the spectrum stands for the strength of variation components. Given that the typical silicon and copper thermal constants are about 2ms, the frequency components ranging from 0 to 1MHz should be mainly contributed by the P and T component. In contrast, the high-frequency components from 1MHz to 250MHz are mainly contributed by the V component, though the frequency boundaries are not necessarily exact.

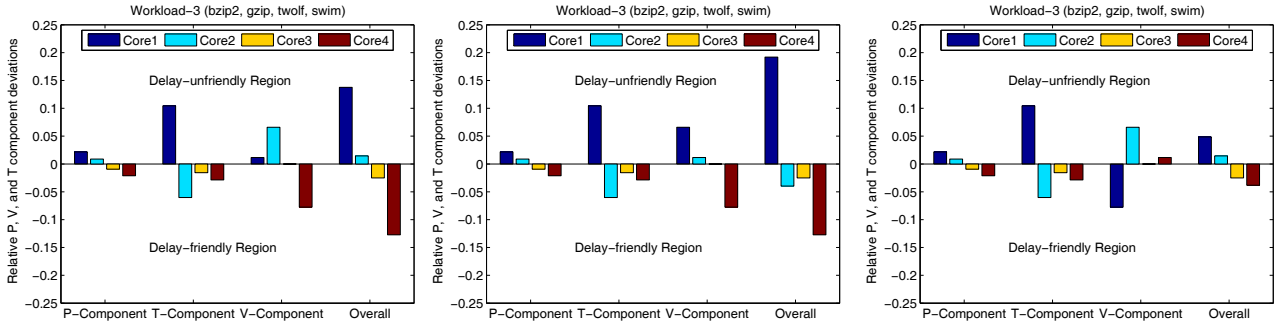
To clearly expose the impacts of the P, V, and T component of each core on a multi-core processor running different applications, we further investigate the frequency spectrum of two quad-core processors running different workloads. To highlight the core-to-core variations, we plot each core's PVT components relatively to their nominal case in Figure 3. The positive deviation implies the variation component increases circuit delay, hence delay-unfriendly, while the negative deviation implies the variation components help reduce circuit delay, hence delay-friendly. We also plot the overall variation strength which is the sum of the three variation components in each core.

To reduce timing emergencies and delay variation, a direct way is to reduce the overall variations strength on each core, which is not easy since the variations are either fabrication or application related. Unless we can change fabrication process or program flow, those variations cannot be reduced. More observations find the overall variation strength differs significantly from core to core. This unbalance provides us the unique opportunity to smooth out the overall variation strength across different cores. As shown in Figure 3(a), Core1 suffers large overall variation strength that will incur lots of timing emergencies. The overall variation strength of core4 is negative showing large delay tolerability. If we can exchange the V component of core4 and core1 on the spectrum, it will result in the overall variation strength shown in Figure 3(b), and both cores now become variation mild. The similar situation applies to Core2 and Core3. To switch V component among cores is relatively easy by existing thread migration technique, since V component is mostly thread-dependant. Although T component is also thread-dependant and should not be affected by thread migration, setting up TM frequency (or TM interval) in between the V and T components will only switch the V component and leave T component intact because the two components locate in different frequency regions.

From the frequency domain analysis, we can draw two important conclusions:

• **Optimization Potential:** The unbalanced variation strength on each core can be smoothed out through thread migration. By exchanging V components, each core can obtain the best P, T and V combination that results in smaller overall variation strength and less timing emergencies. But the potential of the scheme is core and application specific. For example in Figure 3(c), there is not much room to optimize however we switch the V components. Our scheme leverages the intrinsic unbalance among cores. It cannot work if all cores are equally timing risky. Since PVT variations naturally create unbalance in the system, our scheme works for most cases as shown in Section 6.

• **Optimization Strategy:** Knowing the individual P, V, and T component is critical to guide specific optimization strategy. As in Figure 3(b), we switch the V components between Core1/Core4 and Core2/Core3 respectively to keep their low variation strength level. The rational behind the spectrum grafts lies in the spectrum separation. P and T component resides at low-frequency region with the center frequency around 300KHz, while the V component mainly stays in high-frequency region with center frequency around 25MHz. To effectively leverage the complementary effects between T and V, the TM frequency has to be properly set without affecting T component. In another word, we want to keep TM frequency higher than T component to achieve frequency separation. T component is determined by the millisecond thermal constant, which indicates the TM interval to be smaller than millisecond. We will show later in this paper that we can safely set TM intervals that meet the frequency separation while incurs little performance overhead.



(a) Original deviation breakdown (b) Thermal-oriented optimization (c) Timing Emergency-oriented optimization

Figure 4: Thermal-oriented optimization vs. Timing Emergency-oriented optimization

TEA-TM is not a thermal management scheme, though many previous thermal-related schemes use the thread migration technique as well [15]. The basic principle of thermal management for a multi-core processor is to exchange the thread on the hottest core with that on the coolest one, expecting to balance the temperature distribution. But from the timing emergency perspective, such thermal-oriented operation can be misleading. Figure 4 shows the reason. Originally, Core1 is the hottest and Core2 is the coolest. To reduce the timing emergency in the traditional thermal management scheme, Core1 and Core2 will exchange their threads. Figure 4(b) shows the overall variation strength of Core1 after the migration. Unfortunately, the variation strength of Core1 increases significantly because the thread on Core2 happens to be voltage violent at the migration moment, which causes even more timing emergencies in Core1. The fundamental reason behind that is the thermal-oriented migration schemes disregard the V components. In contrast, according to our TEA-TM scheme, exchanging the threads between Core1 and Core4 can yield lower overall timing emergencies, as Figure 4(c) shows.

Hence, TEA-TM scheme is not a simply extension from existing thermal management schemes. Temperature based migration is not always helpful if we cannot setup a proper migration strategy based on the frequency separation of variation sources. Simply migrating hot thread with mild voltage to a cool core may not be optimal. Migrating cool thread with violent voltage to a hot core may introduce more timing emergencies. This discovery differentiates our work from others.

3.3 Time Domain Explanation

We explain the scheme at the time-domain as shown in Figure 5. Core1 has relatively low temperature, but Core3 has higher temperature after a period of execution. Moreover, thread running on Core3 exhibits to be more voltage-violent than the thread running on Core1. Obviously, Core 3 will experience more timing emergencies than Core1. After we exchange their threads, both cores will be relatively relaxed in timing. The idea can be simply explained as to switch the voltage-violent threads to process- and temperature-mild cores to average out the variation impact. PVT variations affect the circuit delay with different time and space span so that their circuit effects may not always aggregate. If the system can detect the real-time P and T conditions of all the cores and V conditions of all the threads, optimizations can be applied to alleviate the total variation impact of the system.

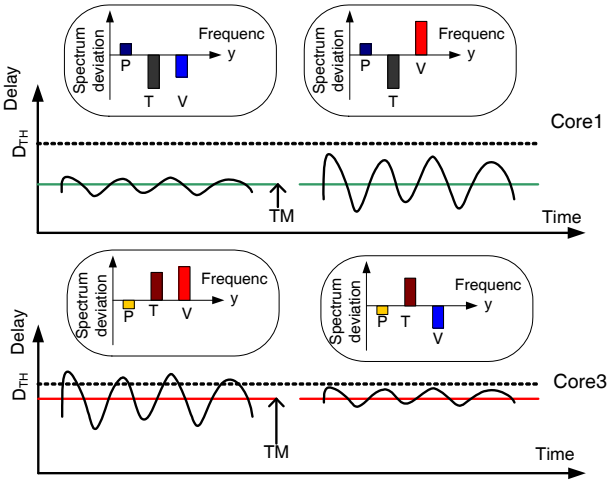


Figure 5: Time-domain explanation of TEA-TM

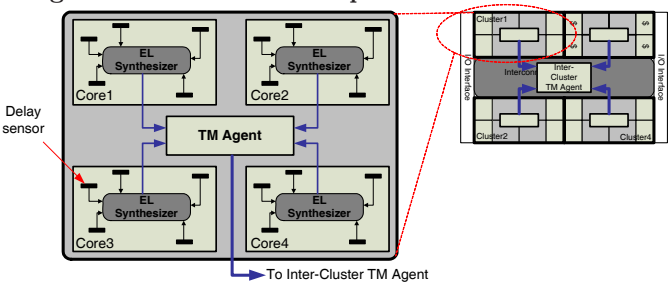
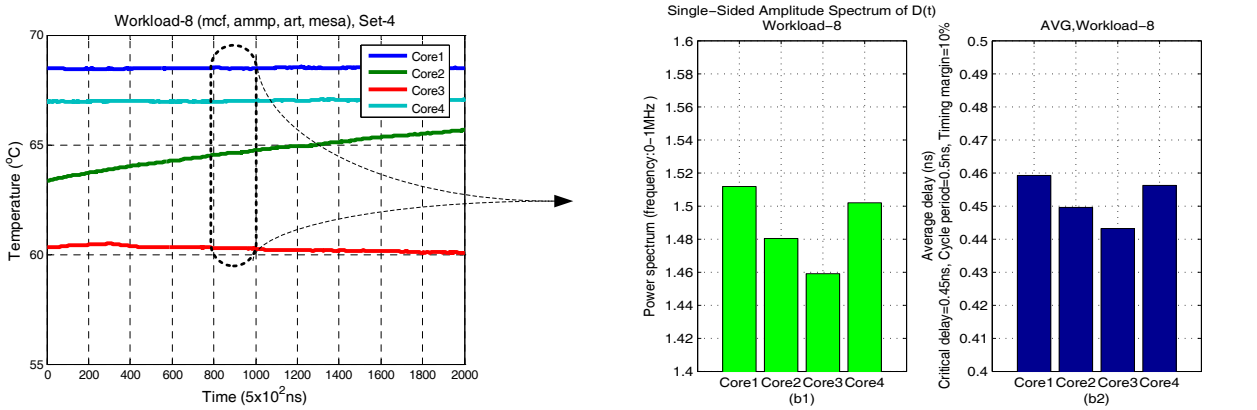


Figure 6: The Framework of TEA-TM

4. IMPLEMENTATION OF TEA-TM

This section discusses three major challenges encountered from implementing TEA-TM and presents several techniques and algorithms to solve them.

Without loss of generality, the implementation assumes a quad-core processor, as shown in Figure 6, which can also be thought of as a typical cluster for future many-core systems [19]. The TM Agent is responsible for generating TM control signals. Multiple delay sensors are deployed into each core to provide accurate and real-time critical delay values. Although these delay sensors faithfully reflect EL, the "raw" delay information is still not enough to guide specific TM strategy. We need to extract corresponding variation strength of P, V, and T components. This brings the first design challenge.



(a) Temperature for four cores

(b) Correlation between temperature and the mean of delay

Figure 7: Using mean of delay values to inter temperature

4.1 Challenge 1: Infer PVT component from Delay Values

Although the frequency analysis in Section 3 can clearly provide the variation strength or each component, the computation and associated storage requirement makes the real-time FFT prohibitively complicated. To reduce the hardware cost, we seek an alternative solution.

Considering that the scheme actually does not need to discriminate between P and T components, because the slow-varying T and static P affect the processor core in almost equivalent manner in a TM interval. We refer P and T component to a unified PT component and discuss how to extract it through delay values.

• **Use mean delay to infer PT component.** Because P and T variations reside in low frequency region (<1MHz), while the delay values cover many random samples with spectrum span up to 250MHz, the arithmetic mean value of all the delay samples serves as a good approximation to reflect the low frequency PT component. To prove this argument, we conduct the following experiment as Figure 7(a) shows. We extract the thermal trace for a quad-core processor running four SPEC2000 benchmarks. In this experiment, we only consider T component since P is simply a DC constant for each core. Within the rectangular period indicated by the dotted lines, Core1 is the hottest, followed by Core4, Core2, and Core3. We conduct FFT on the circuit delay values for the same period. We plot the total variation strength of PT component (below 1MHz) for each core on Figure 7(b1). Comparing to the temperatures of the four cores shown in 7(a), we find that the temperature of each cores is well correlated with the their PT component.

Furthermore, we find that using the mean delay to approximate PT component is very effective. Figure 7(b2) shows the average delay of the same period. We find that the average delay is also well correlated with the core temperature. This greatly simplifies the hardware to extract PT component of each core since calculating the mean delay only needs an accumulator and shifters. This simplification greatly facilitates cost-efficient implementation of TEA-TM.

• **Infer V component.** As pointed out in Section 3, TEA-TM needs two types of information. The variation conditions of the cores are mostly dictated by low-frequency PT component, which can be directly calculated from mean delay. The variation conditions of the thread are mainly related to the high-frequency V components, which cannot be obtained through simple mathematics — this bring us the second challenges. We will propose a greedy approach to avoid the explicit dependence on V component in our TEA-TM scheme in following subsection.

4.2 Challenge 2: On-the-fly TEA-TM Decision Making

Unlike the TM for thermal management where the agents responsible for making TM decisions operate at milliseconds, our TM decision making agent has to finish at small time span requiring more efficient algorithms.

The basic policy is to migrate the most voltage-violent thread to the most process- and temperature-mild cores, as Section 3 explains. However, we cannot directly calculate the V component explained in Section 4.1. We observed that a core associated with small PT component and high EL typically has large V component and tend to be running a voltage-violent thread. In contrast, a core with large PT component but low EL typically has small V component. This observation motivates a decision making policy based on EL and PT component only, and thereby obviating the need for calculating the exact V component.

Consider M cores c_1, c_2, ..., c_M and N threads p_1, p_2, ..., p_N, $N \leq M$. Assume at the start of the kth TM interval, the predicted PT components of the M cores are $PT_1(k)$, $PT_2(k)$, ..., $PT_M(k)$, respectively, and the predicted EL of the N threads are $EL_1(k)$, $EL_2(k)$, ..., $EL_N(k)$. Without loss of generality, we assume that before migration, thread p_i is assigned to c_i, $i = 1, 2, \ldots, N$.

We propose two heuristics to guide the decision making procedure.

• **Urgent First Policy (UFP):** We rank the N threads according to their EL, and sort them with location index $L_{EL} = [a_1, a_2, \ldots, a_N]$. The thread with highest EL is assigned to a_1. For example, $a_1 = 2$ means Thread 2 (p_2) has the highest EL. We also prioritize the M cores according to their PT, and sort them in $L_{PT} = [b_1, b_2, \ldots, b_M]$. The most PT-violent core is assigned to b_1. For example, $b_1 = 3$ means Core 3 (c_3) has the highest EL. The specific relocation strategy can be expressed as migrating thread a_1 to core b_M, thread a_2 to the core b_{M-1}, and so on.

This heuristic is not always optimum because it may waste some cores' tolerability. Assume thread a_1 has the highest EL mainly due to high temperature, but not voltage fluctuation. Switching this thread to PT-mild cores may not be optimum in terms of the overall EL reduction, since the PT-mild core

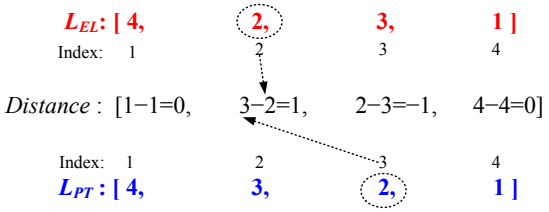


Figure 8: Example: Distance calculation for DUFP

should have been assigned to a voltage-violent thread. This is mostly due to the fact that we cannot directly calculate V component but use EL as an indicator instead.

• **Distance Driven Urgent First Policy (DUFP):** To overcome the disadvantage of UFP, we propose DUFP to further improve the effectiveness. Here, we present an example to clarify the policy. Assume we have $L_{EL} = [4, 2, 3, 1]$ and $L_{PT} = [4, 3, 2, 1]$. In this case, p_4 has the highest EL indicating largest V component. But p_4 is running on c_4 and c_4 has highest EL. This means p_4 might not be the most voltage-violent thread since the high EL might be due to the high PT component on this core. To consider this factor, we define *distance* between L_{EL} and L_{PT}.

For example, c_2 takes the third place in L_{PT}, while p_2 takes the second place in L_{EL}. The distance is calculated as $3-2 = 1$ as shown in Figure 8. Similarly, we can calculate the distance of the other cores. The larger distance implies that the thread is likely to be more voltage-violent, and should be assigned to a PT-mild core. If two cores have the same distance, the thread running on the core with higher EL gets priority. This results in a TM pattern for the next interval as follows:

- Thread 2 will migrate to Core 1;
- Thread 4 will migrate to Core 2;
- Thread 1 will migrate to Core 3;
- Thread 3 will migrate to Core 4.

As for a comparison, the TM pattern of UFP policy for the same case is shown below:

- Thread 4 will migrate to Core 1;
- Thread 2 will stay on Core 2;
- Thread 1 will migrated to Core 4;
- Thread 3 will stay on Core 3.

4.3 Challenge 3: On-the-fly Variation Prediction

The objective of TEA-TM is to reduce the timing emergencies in the future. According to our decision-making heuristics, we need to predict the EL and PT component of the next TM interval based on their historical values. We use a linear prediction mechanism to fulfill this purpose.

The theory of linear prediction is fundamental to many signal processing applications. Least-square method is commonly applied in the linear regression to identify the parameters of the process models [20][21].

Our problem can be expressed as

$$Z(k) = \sum_{i=1}^{M} a_i \times Z(k-i) \tag{4}$$

We predict $Z(k)$ using a linear combination of M most recent past samples. The integer M is called the prediction order. Some training samples are necessary to determine the parameters a_i, i=1,2,..., M. Assuming T training samples

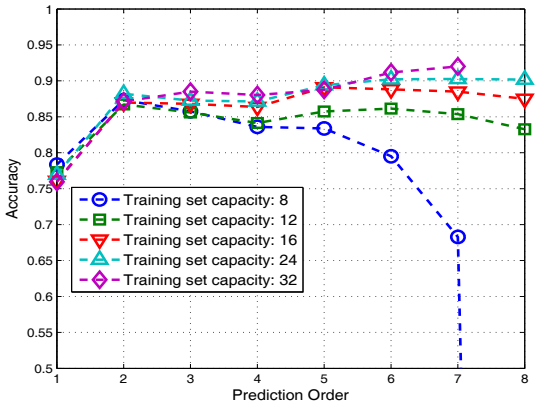


Figure 9: Accuracy vs. Prediction Order and Sample Capacity

are available, i.e $Z(b_1)$, ..., $Z(b_T)$. The following equation can be obtained:

$$Y = XA, \tag{5}$$

where

$$A = [a_1, a_2, \ldots, a_M]^T \tag{6}$$

$$X = \begin{bmatrix} Z(b_1-1) & Z(b_1-2) & \cdots & Z(b_1-M) \\ Z(b_2-1) & Z(b_2-2) & \cdots & Z(b_2-M) \\ \vdots & \vdots & \ddots & \vdots \\ Z(b_T-1) & Z(b_T-2) & \cdots & Z(b_T-M) \end{bmatrix},$$

and

$$Y = [Z(b_1), Z(b_2), \ldots, Z(b_T)]^T.$$

If $X^T X$ is non-singular, the least-squares estimator can be calculated by

$$A = (X^T X)^{-1} X^T Y. \tag{7}$$

When $X^T X$ is singular, $A = \mathbf{1}$ is adopted. The parameter A can be updated with the newly available training samples.

The prediction accuracy is affected by two parameters: prediction order and training size. Our experimental results show that neither high nor low prediction order yields the best prediction accuracy. Figure 9 shows the results of using the simplest one order predictor —a typical "last-value" predictor for EL prediction. The accuracy on average is between 75% and 80%. Two-order predictor can reach up to 87%. Higher orders do not necessarily perform better, probably because the high-order predictors involve too many states that can hurt some locality. Moreover, Figure 9 indicates that higher-order predictors need larger training samples. Overall, a five-order predictor is good enough for EL prediction achieving 90% accuracy.

For PT prediction, one order predictor is sufficient since the P component never changes, and the T component can be thought as unchanged for small TM intervals.

4.4 Hardware Cost

All the implementation above is cost-efficient. We assume the processor has already equipped with the capability of thread migration and rollback checking. Besides delay sensors, we don't need any other sensors. For each delay sensor in the core, two accumulators are implemented: one for recording the EL and the other for calculating the mean delay. We implemented the five-order EL prediction logic with

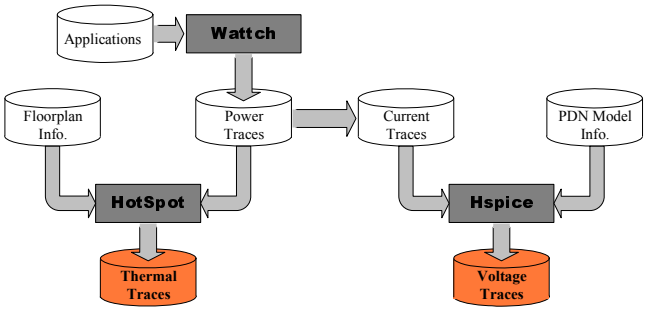


Figure 10: Experiment methodology

Table 1: Processor core configuration

Parameter	Configuration
Clock Frequency	2GHz
Fetch/Issue/Commit	4
Issue Queue/ROB	20/80
Load/Store Queues	64
Functional Units	4-Int/1-cycle latency, 4-FP/7-cycle
Branch Predictor	8K Hybrid Bimodal
L1 I-Cache/D-Cache	64KB, 64B blocks 2-way/4-way, 1-cycle
L2 Cache	2MB, 256B blocks, 8-way, 12-cycle

16 training sets and the DUFP logic in Verilog. The netlist synthesized with Synopsys Design Compiler only consists of several thousands of logic gates. Overall, the hardware cost is negligible.

5. EXPERIMENTAL METHODOLOGY

Figure 10 shows our experimental framework. For each workload, the power traces are generated by Wattch [22]. With a Alpha21246-like floorplan information, we use HotSpot [8] to generate the temperature traces. To get the voltage traces, we first convert the power traces to current traces under a constant voltage level. To get the voltage variation of each core, we use the current traces as stimuli for stressing power delivery network. We use Hspice simulation to expose accurate voltage fluctuations (the simulation time of each workload is about 45 minutes on a 2.33GHz 8-core Xeon workstation).

5.1 Processor Configuration and Workloads

We extend a homogeneous two-core processor used in [14] to a quad-core processor. The processor cores are based on modified SimpleScalar simulator [23]. Each core has private L1 data and instruction caches and L2 caches are shared. Both the L1 data and L2 caches are write-back and write-allocate. The baseline processing core configurations are listed in Table 1.

We use ten mixed workloads from SPEC CPU2000 benchmarks, as Table 2 shows. The workload combinations are similar to that used in [9]. We use SimPoint [24] to sample the simulation intervals based on standard single simulation points configuration. We assign the ten workloads to ten different quad-core processors with each processor suffering different process variation.

5.2 Power Delivery Network

The power delivery networks (PDN) for the modern processors are hierarchically organized. We take a quad-core processor, resembling to Intel Xeon 5500 series processor, as the PDN of our processor. Figure 11(a) illustrates the recommended PDN design for Intel Xeon Processors [25]. The

Table 2: Mixed workloads for quad-core processor

No.	Benchmarks	Property (INT/FP)
1	gcc, gzip, mcf, vpr	int, int, int, int
2	crafty, eon, vortex, vpr	int, int, int, int
3	bzip2, gzip, twolf, swim	int, int, int, fp
4	vortex, vpr, eon , lucas	int, int, int, fp
5	gcc, eon, art, equake	int, int, fp, fp
6	gzip, twolf, ammp, lucas	int, int, fp, fp
7	gcc, applu, mgrid, galgel	int, fp, fp, fp
8	mcf, ammp, art, mesa	int, fp, fp, fp
9	art, lucas, mgrid, swim	fp, fp, fp, fp
10	ammp, applu, mesa, equake	fp, fp, fp, fp

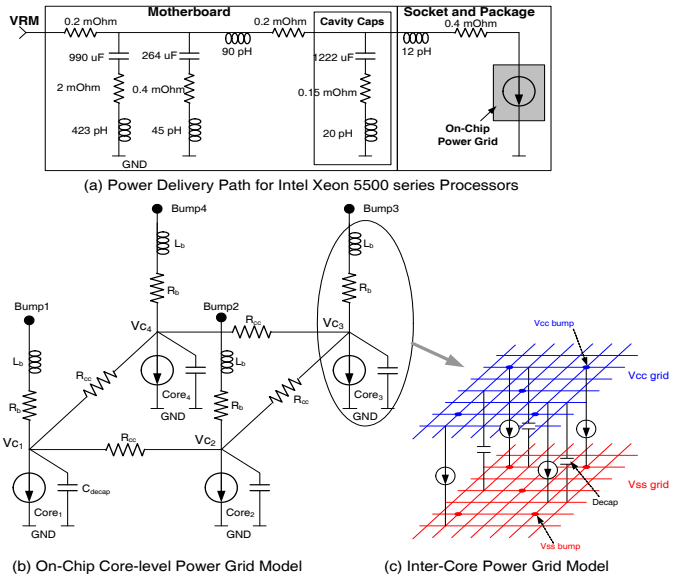


Figure 11: Intel Xeon processor 5500 series-based power delivery impedance model path [25]

power budget is 130W (peak 150W) at the highest voltage level of 1.35V, which is close to the spec of our simulated processors. We use a lumped power grid model for the quad-core processor, as Figure 11(b) shows. To highlight the intra-core power supply interactions and keep the simulation short, we use the following simplification: 1) each core was modeled with a time-varying current source and a decoupling capacitance C_{decap}; 2) the intra-core current paths are modeled with a resistor R_{cc}; 3) multiple voltage bumps serve as the voltage supplier to the cores, through a bump inductor L_b and resistor R_b = $0.1mOhm$. In our PDN model, $C_{decap} = 400nF$, $R_{cc} = 10mOhm$, $L_b = 0.1nH$, which comply with a typical 500-pin flip-chip package.

5.3 Relations between PVT Variations and Circuit Delay

As shown in Eq.(2), we need to obtain experience constants for PVT and delay relations. We conducted a detailed Hspice simulation on ISCAS85 Benchmarks circuits. Figure 12 shows HSPICE results for c880, a representative ISCAS85 circuit. We implemented the circuit using the high-performance version of PTM models [26], with 32nm technology. The simulation results indicate that within the temperature range of 25–105^{o}C, the delay linearly increase by about 1.7 picosecond per degree centigrade (1.7ps/^{o}C). The linear relationship also holds for voltage variation (0.55ps/mV). Similar linear trend is also applicable for process variation [27].

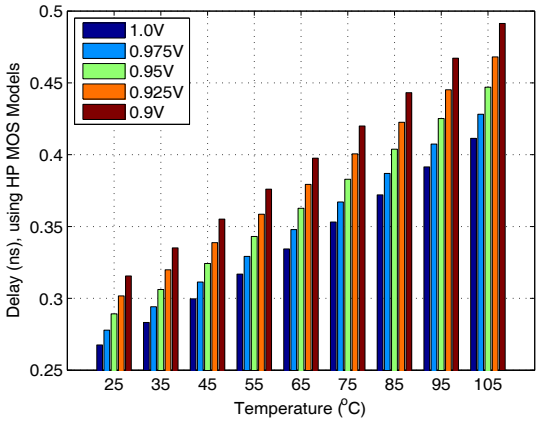


Figure 12: Delay vs (Temperature and Voltage)

Table 3: Parameters used in simulations

Parameters	Values
Timing Threshold	10% cycle period
Process Variation (σ/μ)	1%
Voltage Specification (V_{spec})	1.05V
Temperature Specification (T_{spec})	341K
Frequency	2GHz
Simulation Time	8 million cycles
Wattch Sampling Interval	10 cycles/sample
Hotspot sampling Interval	1K cycles/sample

5.4 Parameter Definitions

Table 3 lists the set of adopted parameters. We assume optimistic 1% process variation and we believe this static variation is relatively easy to compensate using other techniques. Nevertheless, larger process variation can actually improve the effectiveness of our scheme, since larger process variation introduces more unbalance across the cores. The thermal constant is estimated as follows: chip thickness 0.5mm; silicon thermal conductivity $100W/m \cdot K$, copper thermal conductivity $400W/m \cdot K$; silicon thermal capacitance $1.75 \times 10^6 J/m^3 \cdot K$; copper thermal capacitance $3.55 \times 10^6 J/m^3 \cdot K$. The chip's thermal constant should be between 2.2ms and 4.4ms [8].

5.5 Metrics

Higher EL implies higher failure rate and higher performance loss. We assume the performance loss is positively correlated with both EL and IPC for a given thread i, as shown in Eq.(8).

$$P_{loss,i} = \eta \times EL_i \times IPC_i \tag{8}$$

where η is a constant. Based on $P_{loss,i}$, we use relative metric to evaluate the effectiveness of TEA-TM.

Throughput Loss: We define the total Throughput Loss (TL) with N threads running in the processor as the sum of the performance loss of each thread.

$$TL = \sum_{i=1}^{N} P_{loss,i} \tag{9}$$

We define Relative Throughput Loss (RTL) as Eq.(10).

$$RTL = \frac{TL^{\text{w/o TEA-TM}} - TL^{\text{w/ TEA-TM}}}{TL^{\text{w/o TEA\_TM}}} \tag{10}$$

Fairness: TEA-TM leverages the variation and delay unbalance naturally resides in processor cores and always tries to balance them, thereby brings the benefit of fairness across

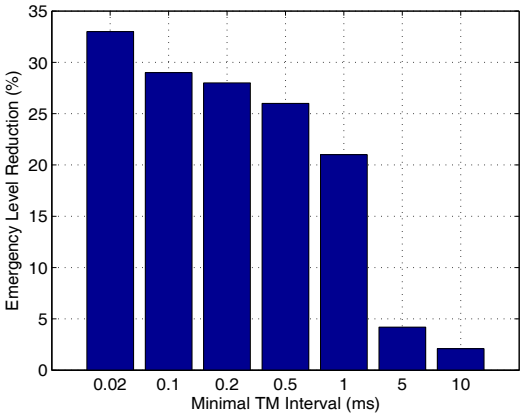


Figure 13: Impact of TM interval on average EL reduction

cores. We use the standard deviation based-metric to evaluate the fairness, defined as

$$Fairness = \frac{1}{\left(\frac{1}{N}\sum_{i=1}^{N}(P_{loss,i} - \overline{P_{loss}})^2\right)^{\frac{1}{2}}} \tag{11}$$

where, $\overline{P_{loss}} = \frac{1}{N}\sum_{i=1}^{N} P_{loss,i}$.

Based on that, we have the Relative Fairness (RF) improvement for the TEA-TM:

$$RF = \frac{Fairness^{\text{w/ TEA-TM}} - Fairness^{\text{w/o TEA-TM}}}{Fairness^{\text{w/o TEA-TM}}} \tag{12}$$

6. SIMULATION RESULTS

6.1 Timing Emergency Reduction

First, we want to investigate the potential of the scheme to reduce EL. The effectiveness is closely related to TM intervals, assuming perfect EL prediction accuracy. In the frequency domain analysis, we have pointed out that TEA-TM only want to switch the high-frequency V component across cores but keeps the low-frequency PT component intact. This means a high TM frequency (or short TM interval) is beneficial for frequency separation. In time domain, this means to find a relatively short TM interval during which the process and temperature cannot change much but the voltage can fluctuate a lot. Figure 13 shows the average EL reduction of ten workloads. We find the EL reduction can reach up to 30% when TM interval is 0.02ms, and is still above 20% with TM interval of 1ms. The effectiveness quickly diminishes with TM increasing to 5-10ms. The result agrees well with the thermal constant (2ms) where TM interval larger than 2ms can no longer separate the V and PT component.

Although a TM interval of 0.02ms can provide the best improvement, this is the ideal case without considering the overhead of thread migration. Previous study shows such frequent TM (60K cycles for 3GHz) can result in about 30% throughput loss [14]. In the later discussion, we adopt a TM interval between 0.1 and 1ms.

Figure 14 shows the EL improvement for 10 workloads. For most cases, TEA-TM reduces the overall EL by a significant amount. However, the potential is workload-specific. The poorest case is for workload 2 — only marginal improvement achieved. This is because all of the threads in workload 2 are high-IPC threads and therefore very power-intensive. This results in high temperature in every core so that there are

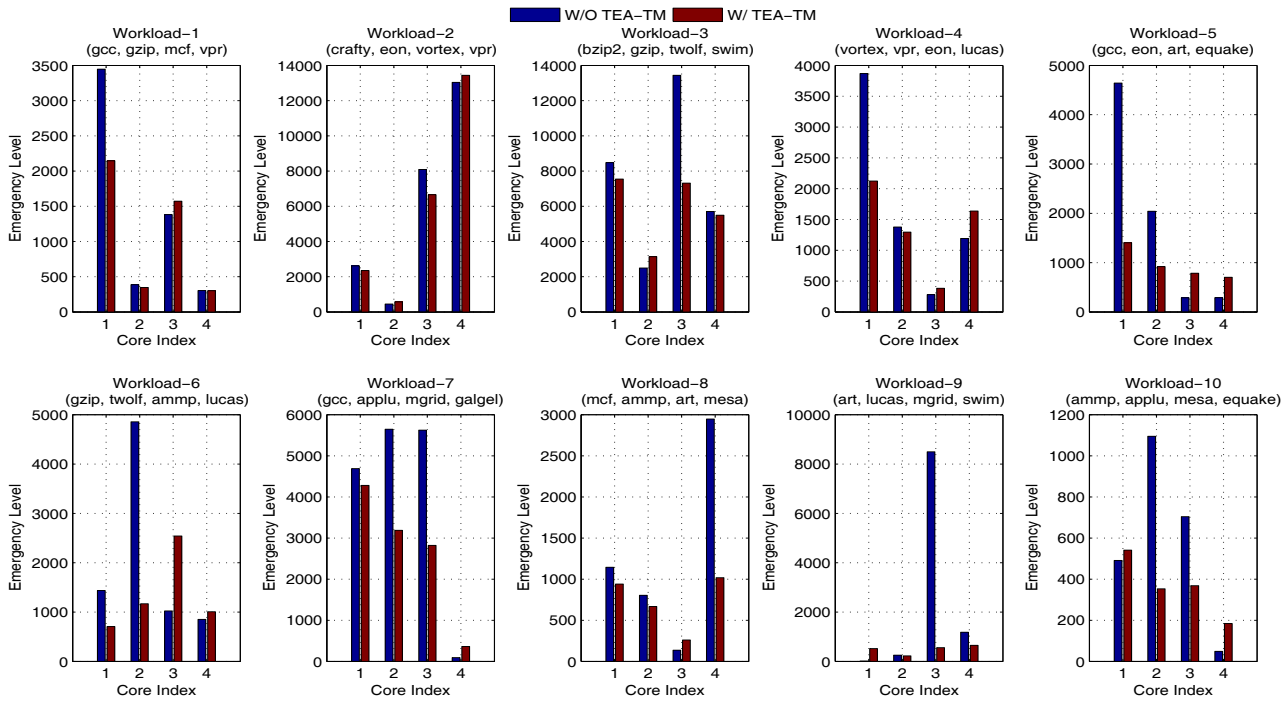


Figure 14: Potential of EL improvement under perfect EL prediction, TM interval: 0.2ms

no mild-cores left for optimization. For other workloads the potential is significant since there are always some mild-cores in the system for tolerating voltage-violent threads.

All the above potential investigation assumes perfect EL prediction with 100% accuracy. We also want to study the impact of imperfect EL prediction. Two types of predictors are evaluated: the simple "last-value" predictor which provides about 80% accuracy and a five-order, 16 training capacity predictor which provides 90% accuracy. Figure 15 shows the percentage EL reduction for different EL prediction accuracies. Even with simpler predictor, we can still achieve meaningful EL reduction from 15% to 25%. The simplest "last-value" predictor can still provide 20% EL reduction with TM interval of 0.2ms. The predictor is barely a register which proves TEA-TM is very cost-effective in hardware overhead. Even a 90% accurate, five-order, 16 training capacity predictor doesn't cost much hardware. Another implication is accuracy matters more for larger TM intervals. Compared with 100% accuracy predictor, the "last-value" predictor degrades about 35% in EL reduction for 1ms TM interval, while it only degrade 26% for 0.1ms TM interval. Therefore, it would be worthy of paying more hardware for accurate predictor when deploying large TM intervals in TEA-TM.

6.2 Relative Throughput Loss Reduction and Fairness

A more strict metric for evaluating the scheme is to use Relative Throughput Loss (RTL) rather than EL since RTL includes the thread IPC information. In this section, we study the RTL reduction of three TM decision-making policies: UFP, DUFP, and Oracle (the hypothetical TM decision-making policy based on predicted EL and requiring post data processing and exhaustive search).

Figure 16 shows that TEA-TM can reduce 22% RTL on average with the simplest UFP policy under 90% EL prediction accuracy. Switching to the more complicated DUFP

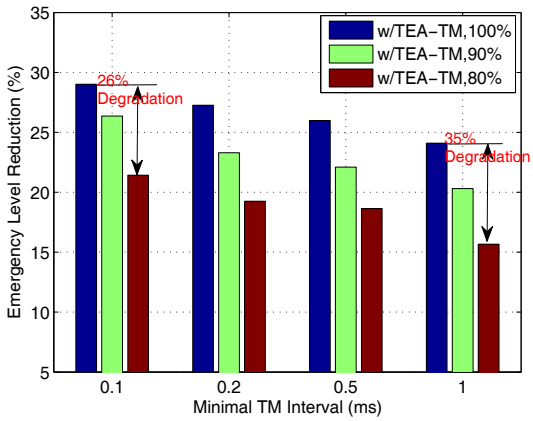
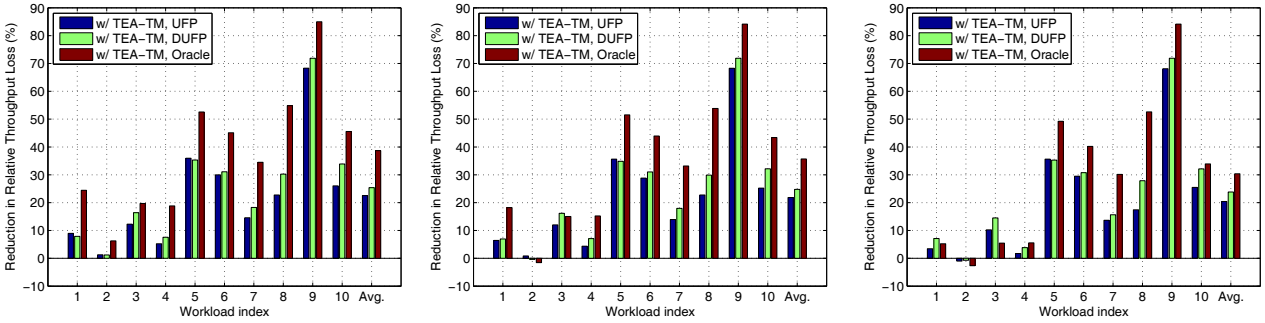


Figure 15: Impact of EL prediction accuracy on average EL reduction

policy adds marginal 3% RTL reduction on average (but for some workloads such as 8 and 10, DUFP can provide decent 7% improvement). Compared with the 35% RTL reduction for oracle policy, there is still much headroom to improve. The large discrepancy between the oracle and the proposed policies lies in the fact that we cannot directly obtain V component. Both policies try to infer V component through EL which can be directly calculated through delay values. Although EL correlates closely with V component, it always carry errors due to other factors. Meanwhile, we find the RTL reduction changes little with different EL prediction accuracies. The RTL reduction only changes from 23% with perfect predictor to 21% with simplest predictor. These observations imply that TM decision making policy is the bottleneck in the current TEA-TM scheme. Developing sophisticated heuristics is more critical than pushing prediction accuracy to higher level.



(a) EL prediction accuracy: 100% (b) EL prediction accuracy: 90% (c) EL prediction accuracy: 80%

Figure 16: Reduction in Relative Throughput Loss. TM Interval: 0.2ms

For workload 2 in Figure 16(b) and (c), TEA-TM generates negative results, even with the Oracle TM policy. This is first because the optimization potential for the workload 2 is very limited. The high IPC of the benchmarks make all the cores equally bad thus eliminates the opportunity for TEA-TM. Also the imperfect predictor may lead to incorrect or unnecessary thread migrations and introduce additional performance overhead. Therefore for the workloads with limited potential, temporary disabling TEA-TM will be a preferred option.

Fairness: Variations naturally introduce EL unbalance among cores. TEA-TM tries to balance the EL across cores through thread migration. In another word, the fairness of the system can be improved. As shown in Figure 14, take the workload 9 for example, the EL of core 3 is eight times larger than the other three cores without TEA-TM. This happens due to a violent thread running on a slow and hot core. If we apply TEA-TM, the burden of running violent thread is shared across the other three mild cores, resulting in all cores' EL kept below a small value. This balance in EL will translate into the fairness of the whole system, as Figure 17 shows. With the UFP and DUFP policy, TEA-TM offers up to 80% fairness improvement on average. For some extreme case such as workload 9, the fairness improvement can be around 400%. The oracle policy can improve 350% fairness on average. In other words, TEA-TM is a fairness-improving scheme which can greatly help smooth the EL unbalance in a multi-core processor. Variations create process, temperature, and voltage unbalance which ultimately lead to delay and timing unfairness in a multi-core architecture. TEA-TM in turn serves as a perfect candidate for such architectures under variations.

7. RELATED WORK

Timing Variation. The implication of device variability on the full-chip timing is discussed in details in [28][10][2] Sarangi et al. studied the possibility of utilizing processors with variation-induced timing errors [29] by trade off the error rate with performance, power, and hardware cost. Gupta et al. also recognized that PVT variations together determine the ultimate timing variation and proposed to use microarchitectural-level recovery mechanism to approach average case design [30]. Razor [31] scheme is another archetype of using delayed error detection to help improve the chips' power efficiency. Unlike their approach, we take the advantage of complementary effects in multi-core processors and only use judicious TM to maximize the tolerability to timing errors.

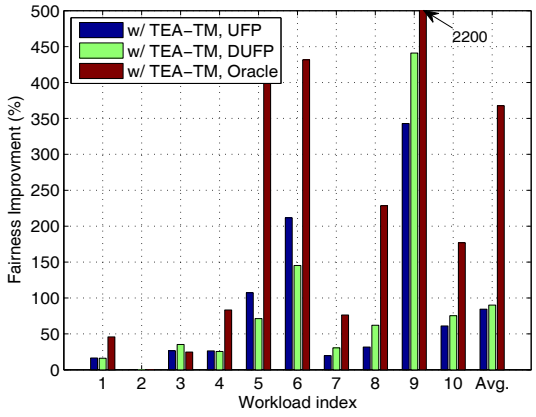


Figure 17: Fairness Improvement. TM Interval: 0.2ms, EL prediction accuracy: 90%

Process Variation. Variable latency techniques for structures such as caches, register files, and pipeline stages can address device variability by tuning architectural latencies [27][32]. Voltage interpolation and fine-grain body bias have also been proposed to mitigate the delay impact of process variation [4][5]. Considering that process variation results in different power profile across cores, the researches proposed to allocate the applications to the cores with lower power consumption, thereby maximizing the power efficiency [33][16]. These approaches, however, is oblivious to temperature and voltage variations and miss the opportunity to leverage the complementary effects to relax the optimization for process variation.

Voltage Variation. A novel voltage variation modeling algorithm was proposed in [34]. Powell et al. proposed the "Pipeline Damping" approach [6] to restrain the rate of current change, thereby indirectly limiting the voltage variation. Mohamood et al. designed an inductive noise controller [35] to alleviate the voltage variation. Recently, Reddi et al. found the voltage emergency can be predicted by monitoring the execution patterns [36], thereby dispensing the voltage sensors which are "notorious" for large response latency. Gupta et al. proposed a "DeCoR" architecture to reactively handle the power noise by engaging a checkpointing and rollback mechanism [7]. Gupta et al. also proposed a software assisted approach [37] to predict voltage emergency through program signatures. These approaches only strive to defend voltage integrity and may be not sufficient to deal with timing emergency as we discussed in Section 2.

Temperature Variation. A temperature model was proposed in [8] and the HotSpot tool motivates many researches on temperature management. Many dynamic thermal management schemes were surveyed in [38][9] [39][40]. Puttaswamy et al. studied the thermal issues in 3D microarchitectures [41]. However, due to the intrinsic thermal sensor latency, the existing thermal management scheme cannot effectively exploit the relatively fine-grained voltage-mild/-violent phase characteristics of a thread due to large operation interval. Although a temperature-balanced situation could be indirectly beneficial to reduce timing emergencies, the aid is probably too far to be helpful.

8. CONCLUSIONS

We proposed TEA-TM, a novel scheme to reduce timing emergencies resulted from PVT variations. Unlike the prior solutions that strive to keep individual P, V, and T in check, TEA-TM leverages the core-level complementary effect between PVT variations. We find that migrating the voltage-violent threads to temperature- and process-mild cores can create variation-mild environment, thereby reducing timing emergency rate. We further offer insights on how the PVT variations can be co-optimized from frequency and time domain perspective. These insights guide to an efficient TEA-TM implementation which only relies on delay sensors. The experimental results show on average TEA-TM can help save up to 24% throughput loss, at the same time improving the fairness by 85% compared with the processor without TEA-TM.

9. ACKNOWLEDGMENTS

The work was supported in part by National Basic Research Program of China (973) under grant No. 2005CB321604, in part by National Natural Science Foundation of China (NSFC) under grant No.(60806014, 60831160526, 60633060, 60921002), and in part by Hi-Tech Research and Development Program of China (863) under grant No.2009AA01Z126.

10. REFERENCES

[1] ITRS, "Process Intergration, Devices, and Structures," 2007 Edition.

[2] O.S. Unsal, J.W. Tschanz, K. Bowman, V. De, X. Vera, A. Gonzalez, and O. Ergin, "Impact of Parameter Variations on Circuits and Microarchitecture," *IEEE Micro*, pp. 30–39, 2006.

[3] X. Liang, D. Brooks, and G.Y. Wei, "A Process-Variation-Tolerant Floating-Point Unit with Voltage Interpolation and Variable Latency," *ISSCC*, pp. 404–405, 2008.

[4] X. Liang, G.Y. Wei, and D. Brooks, "ReVIVaL: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency," *ISCA*, pp. 191–202, 2008.

[5] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "Mitigating Parameter Variation with Dynamic Fine-Grain Body Biasing," *Micro*, pp. 27–42, 2007.

[6] M.D. Powell, and T.N. Vijaykumar, "Pipeline damping: a microarchitectural technique to reduce inductive noise in supply voltage," *ISCA*, pp. 72–83, 2003.

[7] M. S. Gupta, K. Rangan, M. D. Smith, G.Y. Wei, and D. M. Brooks, "DeCoR: A Delayed Commit and Rollback Mechanism for Handling Inductive Noise in Processors," *HPCA*, pp. 381–392, 2008.

[8] K. Skadron, M.R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-Aware Microarchitecture," *ISCA*, pp. 2–13, 2003.

[9] J. Donald, and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration," *ISCA*, pp. 78–88, 2006.

[10] K.A. Bowman, S.G. Duvall, and J.D. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *JSSC*, vol. 37, pp. 183–190, Feb. 2002.

[11] B. Lasbouygues, R. Wilson, N. Azemard, and P. Maurine, "Timing Analysis in Presence of Supply Voltage and Temperature Variations," *ISPD*, pp. 10–16, 2006.

[12] R. B. Staszewski, S. Vemulapalli, P. Vallur, J. Wallberg, and P. T. Balsara, "1.3 V 20 ps time-to-digital converter for frequency synthesis in 90-nm CMOS," *IEEE Trans. on Circuits and Systems II*, pp. 220–224, 2006.

[13] S. Henzler, S. Koeppe, W. Kamp, H. Mulatz, and D. Schmitt-Landsiedel, "90nm 4.7ps-Resolution 0.7-LSB Single-Shot Precision and 19pJ-per-Shot Local Passive Interpolation Time-to-Digital Converter with On-Chip Characterization," *ISSCC*, pp. 548–549, 2008.

[14] T. Constantinou, Y. Sazeides, P. Michaud, D. Fetis, and A. Seznec, "Performance implications of single thread migration on a chip multi-core," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 80–91, 2005.

[15] P. Michaud, A. Seznec, D. Fetis, Y. Sazeides, and T. Constantinou, "A Study of Thread Migration in Temperature-Constrained Multicores," *ACM Trans. Archit. Code Optim.*, vol. 4, no. 2, pp. 1–28, 2007.

[16] K.K. Rangan, G.Y. Wei, and D. Brooks, "Thread Motion: Fine-Grained Power Management for Multi-Core Systems," *ISCA*, pp. 302–313, 2009.

[17] M. Prvulovic, Z. Zhang, J. Torrellas, "ReVive: cost-effective architectural support for rollback recovery in shared-memory multiprocessors," *ISCA*, pp. 111–122, 2002.

[18] D.J. Sorin, M.M.K. Martin, M.D. Hill, D.A. Wood, "SafetyNet: Improving the Availability of Shared Memory Multiprocessors with Global Checkpoint/Recovery," *ISCA*, pp. 123–134, 2002.

[19] M. Tremblay and S. Chaudhry, "A Third-Generation 65nm 16-Core 32-Thread Plus 32-Scout-Thread CMT SPARC Processor," *ISSCC*, pp. 82–83, 2008.

[20] P. P. Vaidyanathan, *The Theory of Linear Prediction*. Morgan & Claypool Publishers series, 2008.

[21] X. Chen, "Recursive least-squares method with membership functions," *International Conference on Machine Learning and Cybernetics*, pp. 1962–1966, 2004.

[22] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *ISCA*, pp. 83–94, 2000.

[23] D. Burger, and T. Austin, "The SimpleScalar Tool Set, Version 2.0," *Computer Sciences Department, University of Wisconsin-Madison, Technical Report 1342*, 1997.

[24] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically Characterizing Large Scale Program Behavior," *ASPLOS*, pp. 45–57, 2002.

[25] Intel, "Voltage Regulator Module (VRM) and Enterprise Voltage Regulator-Down (EVRD) 11.1," *Design Guidelines*, March 2009.

[26] W. Zhao, and Y. Cao, "New generation of Predictive Technology Model for sub-45nm early design exploration," *IEEE Trans. on Electron Devices*, vol. 53, pp. 2816–2823, Nov. 2006. http://www.eas.asu.edu/∼ptm.

[27] X. Liang, and D. Brooks, "Mitigating the Impact of Process Variations on Processor Register Files and Execution Units," *Micro*, pp. 504–514, 2006.

[28] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter Variations and Impact on Circuits and Microarchitecture," *DAC*, pp. 338–342, 2003.

[29] S. Sarangi, B.Greskamp, A. Tiwari, and J. Torrellas, "EVAL: Utilizing Processors with Variation-Induced Timing Errors," *Micro*, pp. 423–434, 2008.

[30] M.S. Gupta, J.A. Rivers, P. Bose, G.Y. Wei, and D. Brooks, "Tribeca: Design for PVT Variations with Local Recovery and Fine-grained Adaptation," *Micro*, pp. 435–446, 2009.

[31] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "A Self-Tuning DVS Processor Using Delay-Error Detection and Correction," *JSSC*, vol. 41, pp. 792–804, Apr. 2006.

[32] A. Tiwari, S. R. Sarangi, and J. Torrellas, "ReCycle: Pipeline Adaptation to Tolerate Process Variation," *ISCA*, pp. 323–334, 2007.

[33] R. Teodorescu, and J. Torrellas, "Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors," *ISCA*, pp. 363–374, 2008.

[34] E. Grochowski, D. Ayers, and V. Tiwari, "Microarchitectural Simulation and Control of di/dt-induced Power Supply Voltage Variation," *HPCA*, pp. 7–16, 2002.

[35] F. Mohamood, M.B. Healy, S.K. Lim, and H.S. Lee, "A FloorplanAware Dynamic Inductive Noise Controller for Reliable 2D and 3D Microprocessors," *Micro*, pp. 3–14, 2006.

[36] V.J. Reddi, M.S. Gupta, G. Holloway, M.D. Smith, G.Y. Wei, and D. Brooks, "Voltage Emergency Prediction: A Signature-Based Approach To Reducing Voltage Emergencies," *HPCA*, pp. 18–29, 2009.

[37] M. S. Gupta, K. K. Rangan, M. D. Smith, G.Y. Wei, and D. Brooks, "Towards a software approach to mitigate voltage emergencies," *ISLPED*, pp. 123–128, 2007.

[38] D. Brooks, and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," *HPCA*, pp. 171–182, 2001.

[39] W. Wu, L. Jin, J. Yang, P. Liu, Sheldon X.-D. Tan, "Efficient power modeling and software thermal sensing for runtime temperature monitoring," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 3, pp. 1–29, 2007.

[40] I. Yeo, C.C. Liu, and E.J. Kim, "Predictive dynamic thermal management for multicore systems," *DAC*, pp. 734–739, 2008.

[41] K. Puttaswamy, and G. H. Loh, "Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High-Performance 3D-Integrated Processors," *HPCA*, pp. 193–204, 2007.