CONTROL TECHNIQUES FOR UNCORE POWER MANAGEMENT IN CHIP-MULTIPROCESSOR DESIGNS

A Thesis

by

ZHENG XU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Jiang Hu
Co-Chair of Committee,	Paul V.Gratz
Committee Member,	Anxiao Jiang
Head of Department,	Chanan Singh

August 2013

Major Subject: Computer Engineering

ABSTRACT

In chip-multiprocessor (CMP) designs, when the number of core increases, the size of on-chip communication fabric and data storage grows accordingly and therefore the chip power challenge is exacerbated. This thesis work considers the power management for networks-on-chip (NoC) and the last level cache, which constitute the uncore in CMP designs. NoC is regarded as a scalable approach to cope with the increasing demand for on-chip communication bandwidth. The last level cache is shared among all cores. The focus of this work is on the control techniques for uncore dynamic voltage and frequency scaling. A realistic but not well-studied scenario is investigated. That is, the entire uncore shares a single voltage/frequency domain, as opposed to separated domains in most of previous works. One appealing advantage here is that data packets no longer experience the interfacing overhead across different voltage/frequency domains. The classic PI (Proportional and Integral) control method is adopted due to its simplicity, flexibility and low implementation overhead. This thesis research outcome includes three parts. First, stability of the PI control is analyzed. Second, a model-assisted PI control scheme is proposed and studied. The model assist is to address the problem that no universally good reference point exists for the control. Third, the windup issue for the PI control is investigated. Full architecture simulations are performed on public benchmark suites to validate the proposed techniques. The result show 76% energy reduction with less than 6% performance degradation compared to constantly high voltage/frequency for uncore.

DEDICATION

To my parents, my aunt, my uncle and grandparents.

ACKNOWLEDGEMENTS

I would like to thank all those who encouraged me and helped me during my study and research at Texas A&M University.

Especially, I would like to thank my family for their ceaseless support, encouragement and endless love. Without which I would never able to complete my master so smoothly.

Also, I would like to extend my heartfelt gratitude to my advisor, Dr. Jiang Hu, who gave me constant guidance as well as warm encouragement throughout this research project. He was always patient, kind and helpful whenever I had questions on my academic life. I could not have completed this thesis and knew power management wonder world without his guidance and generous support. I also would like to thank Dr. Paul Gratz and Dr. Anxiao Jiang for being my committee members, and for their suggestions on this research.

Last but not least, thanks to all my friends and colleagues who accompany with me these years, and also the department faculty and staff for giving me a warm and kind environment during my life at Texas A&M University.

NOMENCLATURE

DVFS	Dynamic Voltage Frequency Scaling
V/F	Voltage/Frequency
NoC	Network-on-Chip
LLC	Last Level Cache
CMP	Chip Multiprocessor
AMAT	Average Memory Access Time
PID	Proportional Integral Derivative

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

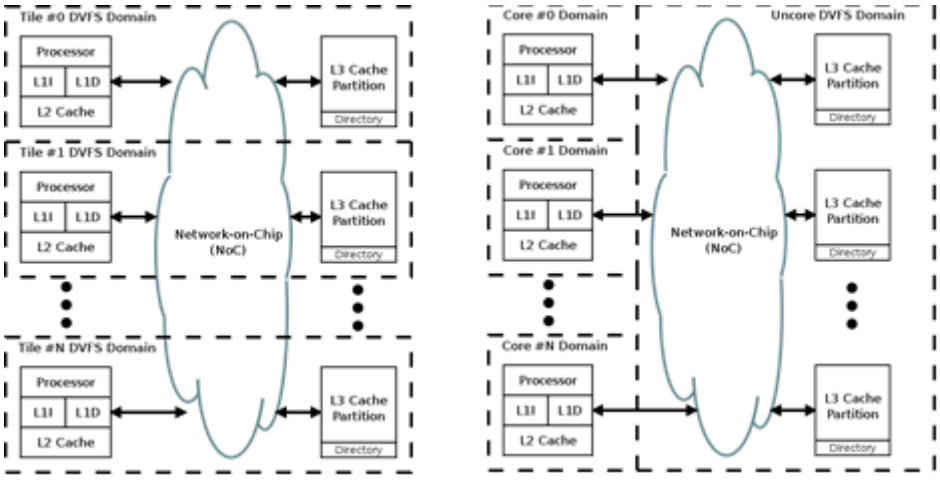
Page

1. INTRODUCTION

Modern chip-multiprocessor (CMP) designs face challenges at multiple fronts. Power is a well-known one that has received a lot of research attention. At the same time, on-chip communication and data storage are becoming a bottleneck to CMP performance. This demands increasingly large bandwidth for the communication as well as bigger on-chip storage size. Indeed, on-chip communication fabrics and cache consume a large portion of chip estate. Evidently, such increase inevitably worsens the power issue. This thesis research is an effort to address these intertwined challenges.

Network-on-chip is an approach for designing the communication subsystem in a CMP (Chip-multiprocessor) system. NoCs apply networking theory and methods to on-chip communication. It brings notable improvements over conventional bus and crossbar interconnections. NoC improves the scalability of CMP, and the power efficiency of complex CMP compared to other designs. Research shows that up to 82% energy savings can be achieved by using NoC compared to conventional bus design in a 16-core system [10]. One study shows that the NoC still takes up a notable proportion of the total chip power which is 36% in MIT RAW architecture [21]. In low power design techniques, when some cores are working in an idle state, they could be allowed to shut down to save power. However, the same idea cannot be used for NoC and LLC since these two parts must function normally even when the workload is low, otherwise the entire CMP could not work at all. Since we cannot shut down NoC and LLC completely, DVFS would be a practical and proper technique for low power design. The target

system will always run at the lowest supply voltage and frequency that meets the performance constraints. This technique has been intensively studied for individual microprocessor cores as well as the NoC [19, 13, 20, 16, 8, 14, 17, 3, 6]. Most of the prior works divide the clock domain based on cores. They put one or more than one core in a voltage/frequency domain along with the nearby distributed cache and interconnection. The divided V/F domains for NoC imply interfacing overhead between different domains as well as significant performance penalty for data communication. The overhead of crossing clock domains would be increased as the number of cores got increased. Our work targets at a realistic scenario where the entire NoC and LLC belong to a single V/F domain. With this architecture configuration, the overhead of crossing clock domain within LLC and NoC can be eliminated. So far, there are only two works [13,6] which are based on the same settings. Liang and Jantsch [13] proposed a rule-based DVFS scheme for NoC using network load as performance metric. Chen et al. [6], developed a PI control based DVFS method with AMAT (Average Memory Access Time) as the metric. They also introduced a low overhead technique for monitoring AMAT.



(a) CMP with V/F domains by tile (b) Separate V/F domain

Figure 1 [6]: Logical CMP diagrams highlighting Voltage/Frequency domain partitions.

This thesis research is focused on the control aspect of the uncore DVFS framework described in [5]. A formal stability analysis for the PI controller is provided. A model-assisted PI control technique is proposed and studied. Experimental results show that it can reduce uncore energy by 74% with no more than 6% performance penalty compared to constantly high uncore V/F level. Furthermore, the windup issue and anti-windup technique for the PI control are studied.

2. BACKGROUND AND RELATED WORK

In this section, we will first introduce the concept of uncore which mainly includes the shared, distributed last level cache and network-on-chip. Next we will talk about the choice of uncore performance metric and the metric monitoring technique. Furthermore, we will also discuss power and performance constraint on uncore. At last, we will introduce the prior related work on DVFS for CMP.

2.1 CMP Uncore Basics

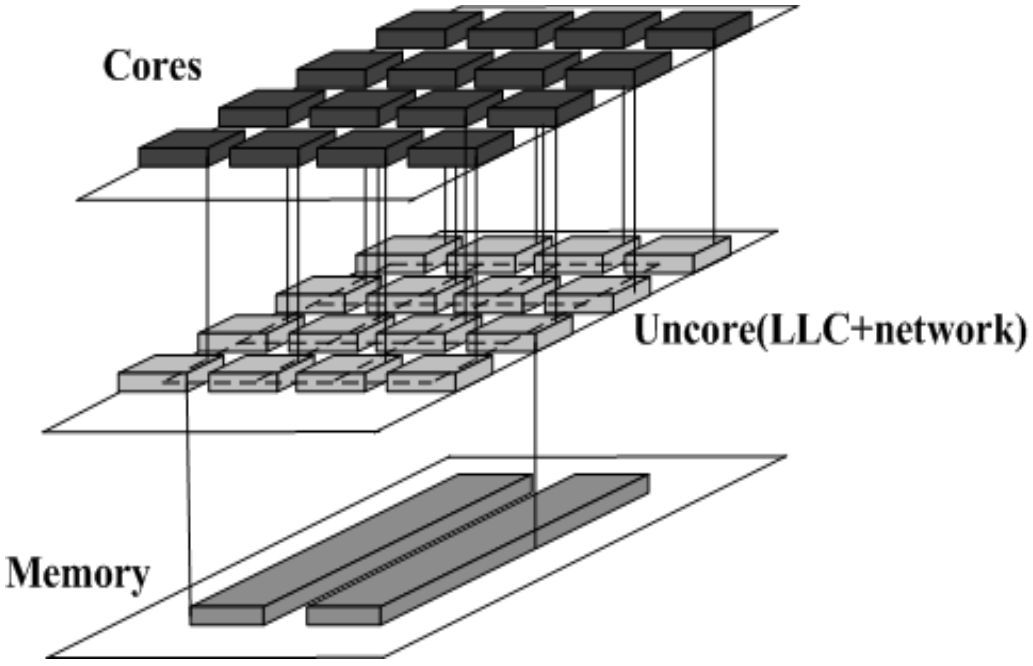


Figure 2 [4]: Core, uncore and memory in a CMP system

In general, CMP contains multicores, private caches and uncore. The uncore portion of the die refers to all the integrated subsystems on the chip except the cores. In more detail, the uncore portion contains the last level cache, the routers and links of NoC, integrated memory controller, integrated I/O controller etc. Uncore is mainly responsible for the communication between core and LLC, on-chip and off-chip. In modern LLCs, LLC is logically shared and spatially distributed. Each bank has an attached core which issues read and write requests. The caches' collective goal is to minimize the use of the main memory.

In the system configuration, it is assumed that coherence between the private caches in the cores is maintained via a distributed directory cache in the uncore. The NoC would carry the memory request from core to LLC, the coherence messages between LLC, the packets between on-chip and off-chip. It is also assumed that LLC cache set indices are spread about the partitions of the LLC in a round robin fashion to ensure that each partition receives approximately the same amount of traffic and no single partition becomes a hotspot.

2.2. Uncore Performance Metric

Liang and Jantsch [13] poposed a DVFS controller which are trying to keep the network workload around its saturation point. The workload here is defined as the number of flits injected into the network. In this way, the V/F level would be adjusted based on the workload of the entire network. If the network workload is considerably higher or lower than the saturation point, the V/F level is adjusted as a respond to the change of workload. There is one apparent drawback in their control mechanism that

their scope only focuses on the total workload of the entire network while neglecting particular congested part of the network. Obviously, a congested part could be able to impact the performance of the entire network. But even with the consideration of local traffic condition of network, it is still not sufficient. The metric they choose to use is the number of flying flits in network which is not appropriate sometimes. Suppose the program contains a large amount of memory store instruction, this will cause a large workload in the point view of number of flits. As we know, memory store instructions are always not as critical as memory load instructions to the overall performance of entire system because memory load instructions have bigger chance to block the execution of following instructions. Thus network workload alone is not a good metric. In Chen et al. [6], a PI control for uncore DVFS is proposed based on AMAT. AMAT is a metric that indicates the memory access's speed of the overall system including private cache access, shared last level cache access, interconnection delay and memory access. AMAT is a global system metric which does not indicate the traffic appropriately all the time. Imagine a common situation, if the program's memory access mostly goes to the off-chip main memory, the AMAT will be very high while the network is actually not busy and there is some room for power saving here. But with AMAT as their metric, the power controller may increase the V/F level based on high AMAT instead of making right decision of decreasing V/F level.

Based on the summary of these two previous work, it is concluded that the ideal metric should reflect both the uncore performance and its criticality to the overall chip

performance. In this research work, the metric proposed in the work of Xi Chen et al. [5] is chosen to use. They proposed a new metric – critical latency which is expressed by

$$\Gamma = \eta \cdot \lambda_U \tag{1}$$

where η is the criticality factor and λ_U is the uncore latency. The uncore latency must consider the latency in both LLC and network. Meanwhile, it must exclude the off-chip main memory access time because it has no relationship with uncore DVFS. Just like the case given in above paragraph, sometimes increasing uncore V/F level does not help to improve the overall system's performance especially when main memory access latency dominates the overall data access latency which is a waste of energy. The uncore latency can be described by

$$\lambda_U = \frac{\left(\sum_{j=1}^{N_{packets}} \lambda_{packet,j}\right) - \lambda_{Mem} \cdot N_{LLC\_Misses}}{N_{packets}} \tag{2}$$

where $\lambda_{packet,j}$ is the total round-trip latency for packet j, $\lambda_{Mem}$is the memory access latency, N_{LLC_Misses} is the number of LLC misses in a control interval and $N_{packets}$ is the number of packets in the same interval.

For memory access instructions, store and load are the two main types. But they have different criticality in terms of the overall system performance. Load instructions in most of the time have much higher priority than store instructions. A load instruction may potentially block the execution of following instructions, but store instruction may not affect the execution of following instruction by using the common techniques such as load bypassing, load forwarding and so on. Thus, the criticality factor of uncore

performance includes *Loads_Fraction*, which is the number of load instructions per cycle. It is expressed by

$$\eta = L1\_Miss \times Loads\_Fraction \tag{3}$$

2.3 Uncore Status Monitoring

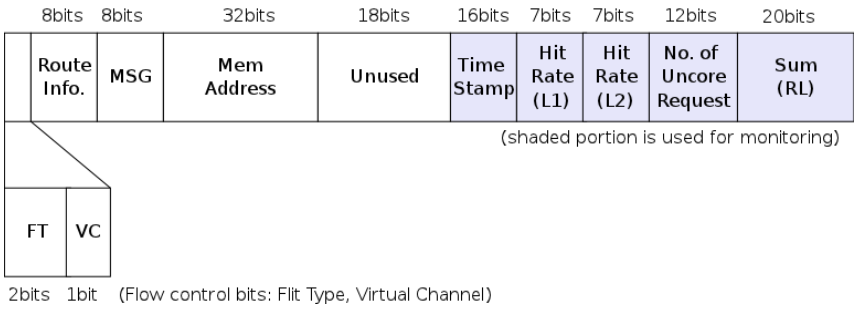


Figure 3 [6]: Header flit bit fields

In Chen et al.'s work [6], they also introduced the implementation of network monitor which has low overhead. Each core keeps their own status information and puts them into every packet header it released to the network. According to the communication policy, the packet header has a large amount of unused bits which make "piggyback" method possible. Xi Chen et al. [6] put a monitor with a central core and collect status information from the header flits passing through.

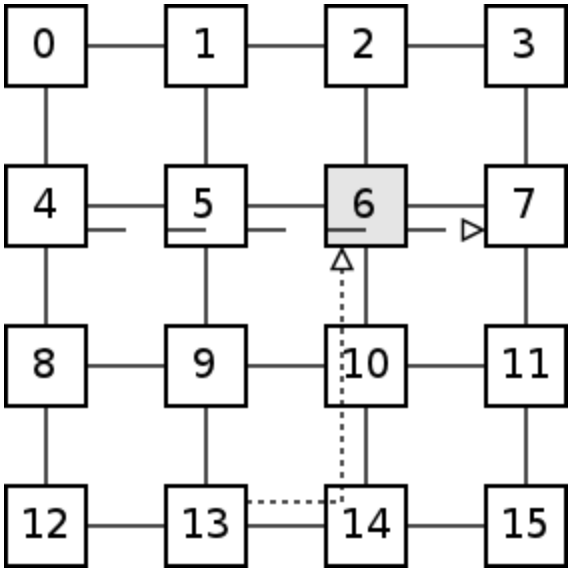


Figure 4 [6]: NoC layout; the monitor resides at tile 6

In the end of each control interval, the data stored in the monitor would be extrapolated based on the arrival time inside the current control interval and averaged across interval among all cores. Figure 5 shows an illustrative example. x_1 and x_2 stand for the data stored in the monitor while N_1 and N_2 stand for the arrival time inside the control interval. In the end of the control interval, x_1 is doubled because the flit from Core 1 arrives in the middle of control interval. The data is averaged across interval among all cores by using the equation in Figure 5. After the extrapolation, the computed metric would be fed into their power controller as input. In this research work, the same method is used since this monitor is very efficient and induces low overhead.

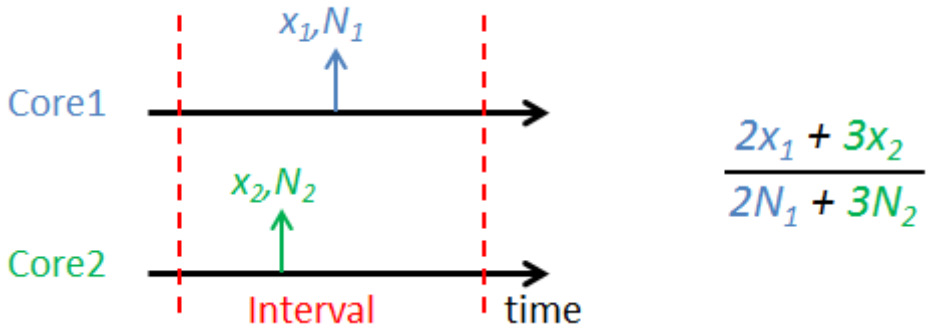


Figure 5: Extrapolation for the rest of the interval

In this thesis research work, we address DVFS for the uncore as a whole. We will show that there is a large opportunity for uncore power saving by using a simple but effective control algorithm to adjust uncore voltage and frequency based on the monitored results.

2.4 PID Control Basics

A proportional-integral-derivative controller (PID controller) is a widely used controller in industrial control systems. It is also called three-term-controller. The popularity is due to its simplicity and efficiency. Moreover, PID controller can keep its performance relatively stable under different conditions of environments. The PID controller contains three separate constant parameters: the proportional, the integral and derivative values, denoted as P, I, and D. In more details, P term depends on the present error, I term depends on the accumulation of past errors, and D term is based on current rate of change of errors. The weighted sum of these three actions is used to adjust the process via a control execution element such as the position of a control valve, a damper,

or the power supplied to a heating element. Sometimes only one or two actions to provide the appropriate system control is sufficient for applications. This is achieved by setting the other parameters to zero. For instance, PI controllers are very popular since derivative term is sensitive to measurement noise. Integral term plays an important role in PI controller because the absence of an integral term may prevent the system from reaching its target value. Xi Chen et al [6] used PI controller in their work. Their reference point in PI controller is a fixed value. This limits the adaptability of the controller. The working condition is changed as the program phase changes so that a more adaptive controller is required to further explore the power saving amount.

2.5 Uncore Power and Performance Implications

In modern chip design, the interconnection portion takes up larger and larger area and power consumption. In the same way, uncore would consume a considerable amount of power so that it is very important to do low power design targeting at uncore portion. Power consumption includes two parts: one is dynamic power dissipation and the other one is static power dissipation. Dynamic power dissipation for CMOS circuit is given by

$$\mathrm{P} = \alpha \cdot \mathrm{C} \cdot V^2 \cdot f \tag{4}$$

where P is the dynamic power dissipation, α is the activity factor, *C* is the capacitance of CMOS, *V* is the supply voltage and *f* is the frequency.

Although the activity factor (α) for uncore is not always high, its total area and capacitance can be large. Static power dissipation for CMOS circuit are given by

$$\mathrm{P} = V_{DD} I_{leakage} \tag{5}$$

where V_{DD} is the supply voltage and $I_{leakage}$ is the leakage current in CMOS.

Although dynamic power takes up largest portion of power dissipation today, static power is increasing very fast and will become a very large part in future VLSI process technologies. From the equation of power calculation, apparently lowering voltage and frequency will bring benefits to low power design directly and significantly. Dynamic frequency and voltage scaling takes advantage of this basic concept and always runs at the lowest supply voltage that meets the timing constraints. On the other hand, lowering the frequency will bring performance degradation in some degrees. But with very careful design of DVFS control policy, the performance degradation could be controlled in a limited range which can be accepted by users. In this paper, a model assisted control technique is proposed which can achieve significant power saving with very limited performance loss.

2.6 Related Works

Several groups have explored DVFS in NoCs and /or CMPs. Shang et al. wrote a pioneering work in the use of DVS for NoCs [19]. They performed DVFS for individual links in NoCs. DVFS has also been studied for individual routers [14]. Mishar et al. changed the V/F level of the upstream router based on the monitored information of input queue occupancy of a router in downstream. Son et al. explored DVFS for specific application in NoCs [20]. They used DVFS on both cores and network links in a parallel linear system solver. V/F levels are picked based on the task criticality. In other previous works, the voltage/frequency domains within the NoC are assumed to be associated with individual processor cores. Guang et al. proposed a voltage island based approach [8]. Individual island V/F levels are chosen according to monitored router

queue occupancies. In each of these works, the DVFS policy is determined by local information. DVFS is also widely studied for processor cores or CMPs. A simple appproach is rule-based DVFS [17] that changes voltage/frequency level when monitored performance crosses certain threshold. Rule-based method is improved by including hysteresis [18]. Control theoretic techniques are proposed in [16, 3]. Orgas et al. [16] employed control theory for voltage island based designs by building state space mathematical models. Furthermore, Bogdan et al. introduced an optimal control method using a fractional state model [3]. In drowsy caches, DVS is employed to reduce the static power by scaling V/F at certain cache lines at a time [7]. So far, there are only few published works on DVFS targeting to shared caches in multicore chips.

3. PRELIMINARIES

3.1 Problem Description

In our system settings, we use a common case in chip multiprocessor design. We simulate a 16-core CMP similar as shown in Figure 6. Each tile consists of a processor with private caches. Each tile also includes a portion of LLC used for both private and shared data depending on the policy implemented by the protocol. The interconnect is a 2D mesh network with routers and network interfaces co-located with every core. The NoC and the LLC together are referred to as uncore system in this paper.

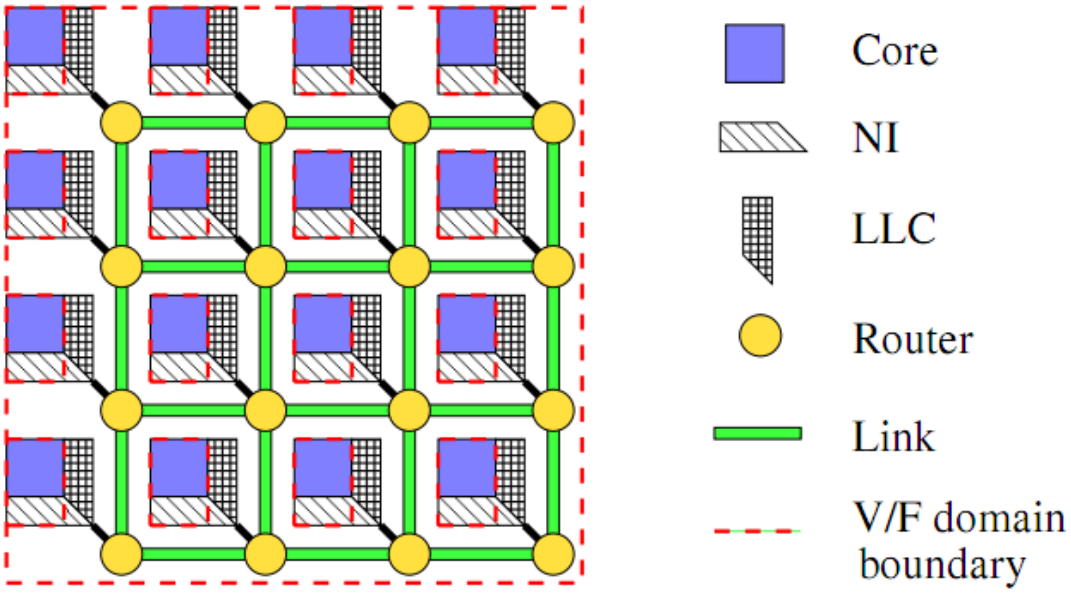


Figure 6 [5]: A multicore processor design where the uncore (NOC+LLC) forms a single V/F domain

The research goal of this paper is formulated as follows. Uncore Dynamic Voltage and Frequency Scaling: for each control interval, use the best control technique to adjust voltage/frequency level for the uncore such that the uncore energy dissipation for both dynamic energy and static energy is minimized while the chip performance for the overall system, in terms of total application runtime, has negligible or user-specified degradation.

3.2 Options for DVFS Control Policy

Generally speaking, there are two main types of approaches for DVFS: open-loop control and closed-loop control. Open loop control does not have a feedback loop, the output is only based on the input signal. Open loop control for DVFS decides control variables based on the current system state and a system model obtained either theoretically or through machine learning. But the CMP system is so complex that the model should depends on a lot of environmental elements. Without the feedback of the controlled system, the reliability and predictability are fragile and tend to be questioned.

Closed loop control compensates this disadvantage of open loop control while brings a little overhead on implementation. Closed loop control compares the measured output signal and the expected output signal and the difference signal would be used to adjust the output signal as controller input. With this closed loop control mechanism, it can be expected to have more accurate control to the system. In the world of closed-loop control for DVFS, we also face a lot of choices such as rule-based, PID (Proportional-Integral-Differential) control, linear control based on state-space model and optimal control. For rule-based control, the uncore V/F level would be decided under a series of

artificial rules. For example, one level would correspond to a range of input variables. For PID control, the decision would be made through the simple, low-overhead PID controller which only relies on a simple mathematical equation. For linear control based on state-space model, the control mechanism would require larger amount of calculation of several matrices. For optimal control, it is the most complex one among control policies. Large amount of calculations have to be executed to generate the final optimized result. With systems get more complicated, the stability is harder to ensure.

In this research, PID is chosen as controller due to its simplicity, flexibility, low implementation overhead and guaranteed stability. But PID is so simple that sometimes it could not achieve the best performance. To compensate this drawback, this thesis work made modifications and proposed a new control mechanism named as model assisted PID control. This new mechanism would not add a lot of complexity while achieved significant performance enhancement. This will be introduced in detail later in subsequent parts of this paper.

4. PID-BASED DVFS POLICY AND STABILITY ANALYSIS

We choose to implement a DVFS control scheme based on PID (Proportional-Integral-Derivative) control. The reason is as follows: First, comparing to rule-based control, the PID control apparently is more flexible to different applications while rule-based control may only work well for some specific applications. Second, comparing to linear control based on state-space model and optimization control, PID controller has both lower computation cost and implementation overhead. Even though PID control is very simple, it has strong theoretic grounds for stability analysis.

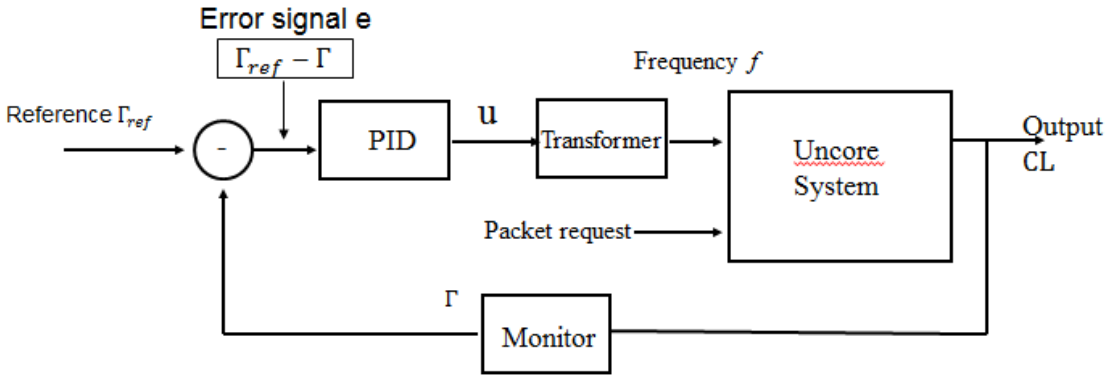


Figure 7: PID system diagram

The block diagram shown in Figure 7 is the PID control system. Let us take a closer look at each part of the system. The controller takes two input signals: the reference critical latency and the monitored critical latency in network. The reference

critical latency is obtained by empirical data. The error is generated by subtracting the monitored critical latency from the reference critical latency which is written as

$$e_i = \Gamma_{ref} - \Gamma_i \tag{6}$$

where Γ_i is the monitored critical latency for the i_{th} control interval and Γ_{ref} is the reference point for the PI controller. Although Γ (critical latency) should correlate with the overall chip performance, its target value is not obvious which need to be decided by empirical data.

The PI controller takes the error as input and it is

$$\mathrm{u(t)} = K_P e(t) + K_I \int_0^t e(\tau) d\tau \tag{7}$$

where K_P is the proportional gain, K_I is the integral gain, u(t) is the output control signal of PI controller which also stands for the uncore clock period and e(t) is the difference between two inputs. From equation (7), the PI controller consists of two items. The first one is proportional control and the second one is integral control. The two control actions in the PI controller are independent of each other. Increasing K_P can result in increasing dynamic response speed of the system and lowering the deviate error of steady state. But if the controller does not contain integral action, proportional control alone can not completely eliminate the steady state deviation. Integral control can compensate this disadvantage and completely eliminate the steady state deviation. As long as there is a deviation, the output of integral part will be accumulated to be larger value and does not diminish until the deviation is totally eliminated to zero. Thus the output will be kept as a constant value. On the other hand of integral control, it will also produce a negative phase shift which means there would be a delay in the response. This

negative effect reduces the stability of the closed-loop system. In engineering applications, according to the controlled object's specific characteristics, workload disturbance and the control performance requirements, the three terms in PID controller can be combined in any way to constitute the necessary control policy. In our case, we are using PI controller since this is simpler and often more robust than including the Derivative (D) term.

Next, make a Laplace transformation to the above equation, we have

$$\mathrm{D(s)} = \frac{U(s)}{E(s)} = K_P + K_I \frac{1}{s} \tag{8}$$

where U(s) is the Laplace transform of u(t), E(s) is the Laplace transform of e(t) and D(s) is the Laplace transform of the PI controller.

Equation (8) is in the form of continuous domain. Considering the reality of our digital system, the ideal continuous integral cannot be achieved in a discrete system. As a result, we have to turn the above equation into a form in digital discrete domain. We will have

$$\mathrm{D(z)} = \mathrm{D(s)}|_{s=1-z^{-1}} = K_P + K_I \cdot \frac{1}{1-z^{-1}} \tag{9}$$

where z comes from Z-transform.

Finally, performing an inverse Z-transform to equation (9), we have the equation below in time domain

$$u_j = u_{j-1} + K_I \cdot e_j + K_P \cdot (e_j - e_{j-1}) \tag{10}$$

where u_j is the output at j_{th} control interval which stands for uncore clock period and e_j is the error at j_{th} control interval.

The control output need to be converted to the V/F level for the uncore system at next. Since critical latency is a nonlinear function with respect to uncore frequency, we perform a transformation of $u = \frac{1}{f}$ such that the critical latency is approximately a linear function of uncore clock period u.

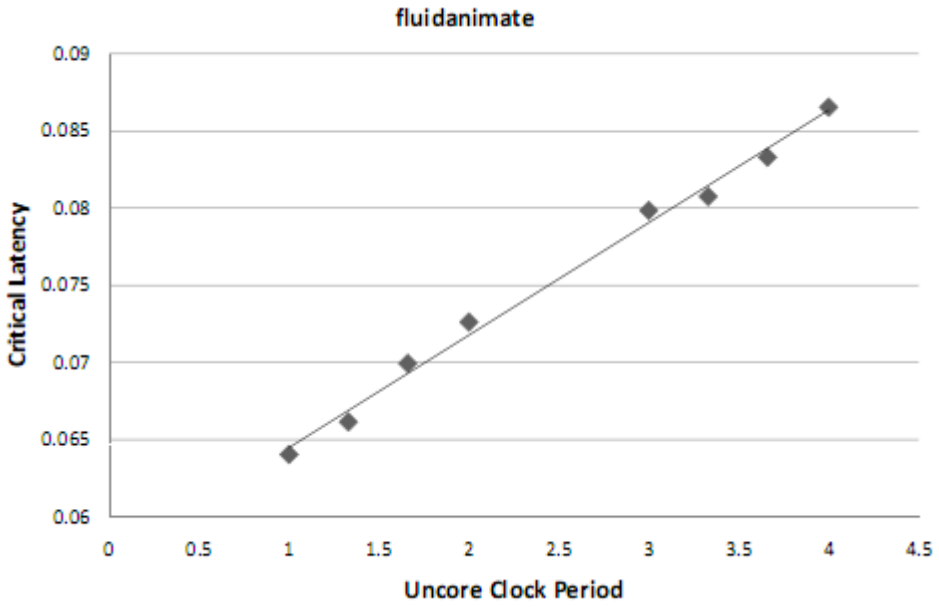


Figure 8 [5]: Critical latency vs. uncore clock period

In order to analyze the stability of this control system, we need to obtain an analytical form of the system function. We tried to obtain some supporting evidence through experiments. For the PARSEC benchmark *fluidanimate*, we simulate with different constant uncore V/F levels throughout the entire ROI (Region of Interest). The

average critical latency versus uncore clock period results are plotted in Figure 8. The results confirm that critical latency has approximately linear dependence on the uncore period.

Then, by curve fitting, we can obtain an approximated expression

$$\Gamma = \alpha \cdot T_U + \beta \tag{11}$$

where $T_U = u$ which is the uncore clock period, α and β are two fitting coefficients. Although the values of α and β are specific for each application, the subsequent analysis is general, as long as the $\Gamma(T_U)$ relation conforms to above equation. We performed many other simulations and all results follow similar trend as the above figure.

For the monitor part, we simply use 1 to represent it as the monitored critical latency is nearly equal to actual critical latency. This has been confirmed in Xi Chen, et al. [6].

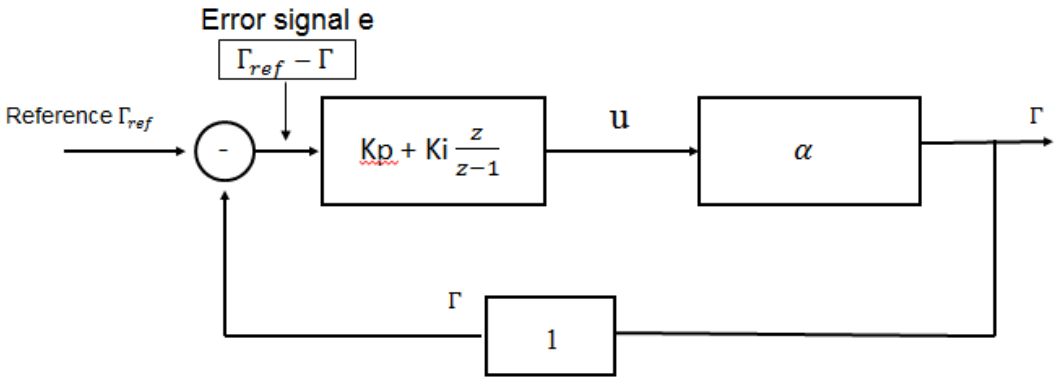


Figure 9: Mathematical PID system model

Combining all the equations together, we have

$$\begin{cases} \Gamma(z) = e(z) \cdot (K_P + K_I \frac{z}{z-1}) \cdot \alpha \\ e(z) = \Gamma_{ref}(z) - \Gamma(z) \end{cases} \tag{12}$$

After substitute the second equation into the first equation, we have

$$\Gamma(z) = \frac{(K_P + K_I \frac{z}{z-1}) \cdot \alpha}{1 + (K_P + K_I \frac{z}{z-1}) \cdot \alpha} \Gamma_{ref}(z) \tag{13}$$

which can be converted to a simplified form

$$\Gamma(z) = \frac{(K_P + K_I) \cdot \alpha \cdot z - K_P \cdot \alpha}{(1 + K_P \cdot \alpha + K_I \cdot \alpha) \cdot z - (1 + K_P)\alpha} \Gamma_{ref}(z) \tag{14}$$

The characteristic equation for this system is

$$(1 + K_P \cdot \alpha + K_I \cdot \alpha) \cdot z - (1 + K_P)\alpha = 0 \tag{15}$$

Stability analysis is a very important part of control theory. In our case, if the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen poorly, the controlled process can be unstable. The system stability is defined for situations where its equilibrium state experiences external disturbance. When the disturbance disappears after a period of time, if the system can go back to the original equilibrium state, the system is called stable system. In contrast, if the system can not return to the original state of equilibrium, the system is called unstable system. Linear stability of the system is the inherent characteristic which only depends on the system itself regardless of the presence and strength of the external input signal.

According to control theory, the system is stable if and only if the root of the characteristic equation is inside the unit circle of the z-plane. In our case, this requires that

$$z = \frac{(1+K_P)\cdot\alpha}{1+K_P\cdot\alpha+K_I\cdot\alpha} \tag{16}$$

is within the unit circle. To satisfy this condition, we simply need to find PID coefficients K_P, $K_I > 0$.

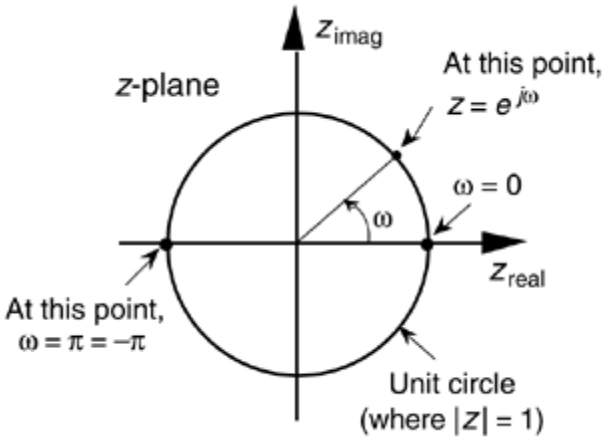


Figure 10: Unit circle in Z-plane

When there is no miss request issued into the network, the critical latency is not affected by f and the above closed-loop based analysis does not hold. However, the chance of no miss request to LLC is very small. Even when it happens, the monitored critical latency is very low and the controlled frequency gradually decreases to its minimum. Therefore, the system is still stable.

5. MODEL ASSISTED PI CONTROLLER

Both Liang and Jantsch's [13] and Chen et al.'s [6] work have a common weakness which is that they have no systematic approach to decide the reference point for their controllers. In Chen et al.'s work [6], the reference point is set empirically based on offline simulations. However, it is very difficult, if not possible, for a fixed reference point to work well for different applications.

The error function in the critical latency based PI controller is

$$e_i = \Gamma_{ref} - \Gamma_i \tag{17}$$

where Γ_i is the monitored critical latency for the i_{th} control interval and Γ_{ref} is the reference point for the PI controller. Although Γ should correlate with the overall chip performance. Its target value is not obvious. Of course, one may select one empirically from offline simulations. However, the offline test cases might not behave the same as online cases.

To address this problem, we propose a model assisted PI control method. We can see that Γ is approximately a linear function of packet latency, which is in turn proportional to the uncore clock period T_U. Hence, we have

$$\Gamma = \alpha \cdot T_U + \beta \tag{18}$$

where α and β are coefficients independent of T_U. Within each program phase, the program execution behaviors are generally consistent so that the variations of α and β are often limited. Based on $(T_{U,i-1}, \Gamma_{i-1})$ of the previous interval, we can estimate the values for α and β. Then, we can predict the $\Gamma(T_U)$ function of the next control interval.

This prediction can guide reference V/F to a more aggressive or more conservative level. We first find three different reference points empirically, one is normal, another one is aggressive and the other one is conservative. Then, we dynamically choose among them at runtime according to the uncore frequency $\frac{1}{T_{U,i+1}}$ computed from the predicted $\Gamma(T_U)$ function. By default, the normal reference point is employed. If the model based frequency is significantly higher (lower), the reference point is changed to the aggressive (conservative) one. If there is no frequency change in two consecutive control intervals, we cannot obtain an update on α and β values. In this case, we continue to use the reference of the previous control interval.

To show more details about the new control mechanism, the complete algorithm is listed below

Step1: Record $T_{current}, T_{previous}$ which are the clock period of current and previous control intervals, $\Gamma_{current}, \Gamma_{previous}$ which are the critical latency of current and previous control intervals.

Step2: Update the mathematical model coefficients of α and β in the model

$$\Gamma = \alpha \cdot T_U + \beta \tag{19}$$

According to curve fitting, we can update the coefficients by using the following equations

$$\begin{cases} \alpha = \frac{\Gamma_{current} - \Gamma_{previous}}{T_{current} - T_{previous}} \\ \beta = \Gamma_{current} - \alpha \cdot T_{current} \end{cases} \tag{20}$$

Step3: With the updated model, we can substitute T_U with the minimum clock period and calculate the estimated minimum critical latency as follows

$$\Gamma_{minimum} = \alpha \cdot T_{minimum} + \beta \tag{21}$$

Then add an inflation coefficient ρ to get the inflated critical latency,

$$\Gamma_{inflated} = (1 + \rho) \cdot \Gamma_{minimum} \tag{22}$$

where ρ is a small number, e.g., 10% . One can use a greater value of ρ to achieve more energy savings but at the higher performance cost. By magnifying $\Gamma_{minimum}$, the controller moves the V/F level down to the lowest value without significantly hurting the performance of the overall system. The value ρ is set to 10% in our experiment for the best energy-performance trade off.

Step4: In this step, we calculate four clock periods through different methods. The first clock period $T_{model-assist}$ is based on the $\Gamma_{inflated}$ and the model we built in Step2.

$$T_{model-assist} = \frac{\Gamma_{inflated} - \beta}{\alpha} \tag{23}$$

The next three clock period $T_{aggressive}, T_{normal,} T_{conservative}$ are based on three different reference critical latency and PI controller. As we talked in previous chapters, PI controller take two inputs, one is reference critical latency, another one is monitored critical latency. We have

$$\begin{cases} T_{aggressive} = PI(\Gamma_{aggressive\ reference}, \Gamma_{monitored}) \\ T_{normal} = PI(\Gamma_{normal\ reference}, \Gamma_{monitored}) \\ T_{conservative} = PI(\Gamma_{conservative\ reference}, \Gamma_{monitored}) \end{cases} \tag{24}$$

where the three different reference critical latency are chosen by empirical data from a large amount of experiments.

Step5: this is the last step of the control mechanism. In this final step, we choose the clock period from $T_{aggressive}, T_{normal,}\ T_{conservative}$ as three candidates based the comparison result between $T_{model-assist}$ and each of these three candidates. Generally speaking, the final T_U will be the one which is closest to $T_{model-assist}$.

The reason that we do not directly use $T_{model-assist}$ is based on our practical experiment. We discovered that with $T_{model-assist}$, the performance is not as good as we expected. In the procedure of calculating $T_{model-assist}$, there are too much noise that the predicted model sometimes deviates from the actual behavior very far. Hence, we bring in $T_{aggressive}, T_{normal,}\ T_{conservative}$ generated by PI controller and pick one as the final output control signal instead of directly using $T_{model-assist}$.

6. ANTI WIND-UP ISSUE OF PI CONTROLLER

When error between the set point and the feedback value stays in one direction, controller output keeps rising due to the cumulative effect of integral term. This may causes the controller output to reach the upper limit of control actuator. If the direction of the error changes at this moment, the controller output should gradually fall back and the actuator should escape from its saturation; otherwise, the controller output should continue to increase and the actuator should remain at its upper bound. Hence, the actuator enters its saturation zone. The deeper it enters, the longer time it takes to exit. If the error reverses its direction, the actuator would not immediately react. Instead, controller output gradually decreases and the actuator responds with a certain amount of delay. This situation makes control performance degraded and it is referred as integral windup in PID control.

In our research, we prevent the wind-up problem by limiting the integral term between pre-determined lower and upper bounds.

All actuators have saturation limit, i.e. a maximum limit and a minimum limit. For example, a power, and a valve can not have an infinitely large opening and can not be more closed than closed. Under normal process operation the control variable should not reach the saturation limits. In our uncore system, we also have our actuator's constraint. The upper limit and lower limit are the top and bottom V/F levels that we can choose. In our case, the upper limit of frequency is as same as the core frequency and the lower limit is 0.2 times of core frequency. The actuators mentioned in above theories

correspond to the V/F adjusters in our system such as voltage level converter, phase lock loop and so on.

Assume that in a time interval a large amount of memory requests flood into uncore system so that the system's output variable (critical latency) is increased significantly. The control error which is the difference between reference critical latency and monitored critical latency then becomes large and negative, and the control variable which stands for the uncore clock period will decrease (because of the integral term of the PID controller) until the control signal reaches at its minimum value. When the uncore clock period as same as the core clock period, T_{min} is still not small enough to compensate for the large disturbance of uncore workload. Because of this, the control error keeps going large, and the integral of the control error continues to increase, which means that the calculated integral term u_i continues to increase.

Assume that the disturbance lasts for a moment and then disappears, i.e., the uncore workload reduces to a low level. This causes the clock period to increase since the workload is reduced (or the load is removed), and the error will now change sign (it becomes positive). Consequently the integral term starts to integrate downwards (its value is continuously being reduced). So that the calculated u_i is reduced since the smaller amount of workload requires a larger control signal which stands for larger clock period (lower frequency of uncore system). However, the problem is that it may take a long time until the large value of the calculated u_i reduced (via the down-integration) to a normal (reasonable) value. During this long time, the clock period is smaller than what is required to serve the current amount of workload, causing the critical latency to be

lower than the set point during this time. This is counted as wasting energy since there is room for us to lower frequency to achieve power saving. But due to wind-up, the system reacts very slow and loses the opportunity to save power.

To sum it up: a large and long-lasting process disturbance which forces the control variable (via the controller) to one of its saturation limits, implies a long-lasting error different from zero.

A practical PID controller must be able to cope with the possibility of integrator wind-up, that is, it must have some anti wind-up mechanism. The anti wind-up mechanism is always simple by making rules. Since the problem is that the integral term increases continuously during actuator saturation, the solution is to halt the integration when the control signal reaches either its maximum or minimum limit. For the implementation in our PI controller,

$$\begin{cases} e_i = \Gamma_{ref} - \Gamma_i \\ u_i = base + U_P + U_I \\ U_P = K_P \cdot e_i \,,\; U_{I_i} = K_I \cdot e_i + U_{I_{i-1}} \end{cases} \tag{25}$$

where $base$ is an offset value and other values are defined as same as above paragraphs. We set a range for the integral term as follows

$$-th < U_I < th \tag{26}$$

where *th* is a parameter we set for the PI controller. This bound decides the tolerable range that the integral control part can accumulate its history values. Large bound value will make the system react slowly because it will take longer time for the controller output value to go back to the normal range out of which the actuator can only stay at its upper or lower bounds without any change. On the other hand, low bound value may

make the system react very quickly due to the opposite reason. Over react may make system' steady state error to be raised which means the system may always oscillate around the set point instead of settling down exactly onto the set point. Based on the analysis above, we set different value for *th* and make our choice based on the performance in actual benchmark. Finally, we picked 2 as our bound. The detail results will be showed in later chapters.

7. DESIGN IMPLEMENTATION

The proposed implementation techniques mainly include three parts:

(1) All tiles keep their own status information and refresh them in the end of every control interval

(2) It requires a central controller that stores the collected data from each tile and performs computation based on the control algorithm

(3) The communication policy for information transportation from tiles to the central controller

For the critical latency based DVFS control, each tile need to have several registers to save relevant information. The bit-width of each register is decided by the range of the data to be saved. Particularly, we show a design based on our experiment settings. Three 16-bit registers are required to save the number of load instructions, private cache hits and private cache misses, respectively. In addition, there is also a 20-bit register to save the accumulation value of the total request latency which is the period of time from the time of issuing request to the time of request served in network. Furthermore, to count the number of LLC misses, we also require another 12-bit register. At last, there is another 16-bit register required to keep track of control interval. In summary, 112 bits of registers are required for each tile.

In modern multicore processor designs, e.g., Intel's Nehalem architecture [11, 12], there is a Power Control Unit (PCU), which is a small processor dedicated to chip power management. According to the structure of this architecture, we can use PCU to

implement our DVFS control policy without need of help from additional hardware. The PCU retains a lookup table with each entry containing the data for each tile. In addition, the reference critical latency values and parameters for PI controller are stored in PCU. Typically, this PCU has plenty of storage space and sufficient speed of computation to meet all the needs. At the end of each control interval, the PCU would compute the critical latency and uncore V/F level. This computation takes several arithmetical operations and thus can be finished very quickly.

For the transportation of status information from each tile to the central controller PCU, we use a method as same as [6]. When a packet is sent out from a tile, the 96-bit information (by excluding the control interval counter) is scaled to 64 bits and is embedded in the header flit. When the flits passing by the tile where the PCU is located, the data is scaled back to 96 bits and extracted to the lookup table. Within each control interval, new data from a tile overwrites the old data from the same interval. In the procedure of data transmission, we do not use any extra hardware so that with this communication protocol the overhead is fairly low. The work of [6] shows that a single monitor tile can obtain sufficient sample data in a control interval of 50K clock cycles.

Parameter	Values
Core Frequency	1GHz
#processing cores	16
L1 data cache	2-way 256Kb, 2 core cycle latency
L2 cache (LLC)	16-way, 2MB/bank, 32MB/total, 10 uncore cycle latency
Directory cache	MESI, 4 uncore cycle latency
Memory access latency	100 core cycles
NoC	4×4 2D mesh, X-Y DOR, 4 flits depth/VC
Voltage/Frequency	10 levels, voltage: 1V–2V, frequency: 250MHz–1GHz
V/F transition	100 core cycles per step

Table 1: Simulation setup

The simulation setup is shown in Table 1. The baseline architecture in our experiments is a chip multiprocessor which contains 16-tiles. Each tile is composed of a processing core with 1-level of private cache, a network interface, an NOC router and a partition of the shared L2 cache (LLC) and directory. Each core is an in-order processor based on Alpha ISA. All the experimental configurations and parameters are listed in the table above. We run our experiment on Gem5 [2] full system simulator with PARSEC shared-memory multi-processor benchmarks [1]. For each benchmark, the entire application runs in full-system mode; the results obtained are based upon statistics from the region of interest (ROI), which is usually hundreds of million cycles long. Gem5 "Ruby" memory hierarchy (L1+LLC+directory) and "Garnet" network simulator are also used inside the full system simulator. Frequency scaling for uncore is implemented

in the way of changing the latency of each uncore component. For example, the latencies of each router pipeline stage, link traversal time and the LLC access time are doubled if the frequency is reduced to 50% of core frequency. Since uncore DVFS has no impact on off-chip memory access latency, its access latency is treated as a constant value. We expect the proposed techniques should work well with core DVFS as this would be expressed as decreased L1 demand and decreased the utility of the uncore. The control interval for the uncore DVFS is set to be 50K core clock cycles and this choice is made based on a large amount of experiment. According to our experience and existing literature, such interval size allows enough time for the uncore V/F change to settle, and is sufficiently small to capture fine-grain program phase behavior to make quick reaction. Both dynamic and leakage power are considered in the experiments. ORION 2.0 [9] and CACTI 6.0 [15] based on 65 nm technology are used as the power models of NoC and LLC, respectively. The overall performance is evaluated as the execution time for the ROI of each application.

In this work, we compared the following methods:

Baseline: constantly high uncore V/F level

AMAT+PI: Chen, et al.'s method [6]

CL+PI: PI control based on the critical latency described in previous chapters

CL+ModelAssist: the critical latency-driven, model assisted PI control described in previous chapters

8. EVALUATION

In previous chapters, I mentioned about the difficulty to estimate the system's behavior since it is too complex to analyze with accurate mathematical model. Figure 11, Figure 12 and Figure 13 below show some simulation results for a segment of x264 which is one of the benchmarks in PARSEC suites. The horizontal axis stands for the number of control intervals where each control interval is set to be 50K core cycles long, i.e., each data point is an average over 50K clock cycles. The data shows drastic changes from interval to interval from which it is hard to find the rules or regular characteristics. Based on these detail simulation intermediate data, we can see it is very difficult to accurately predict the behavior of a multicore system.

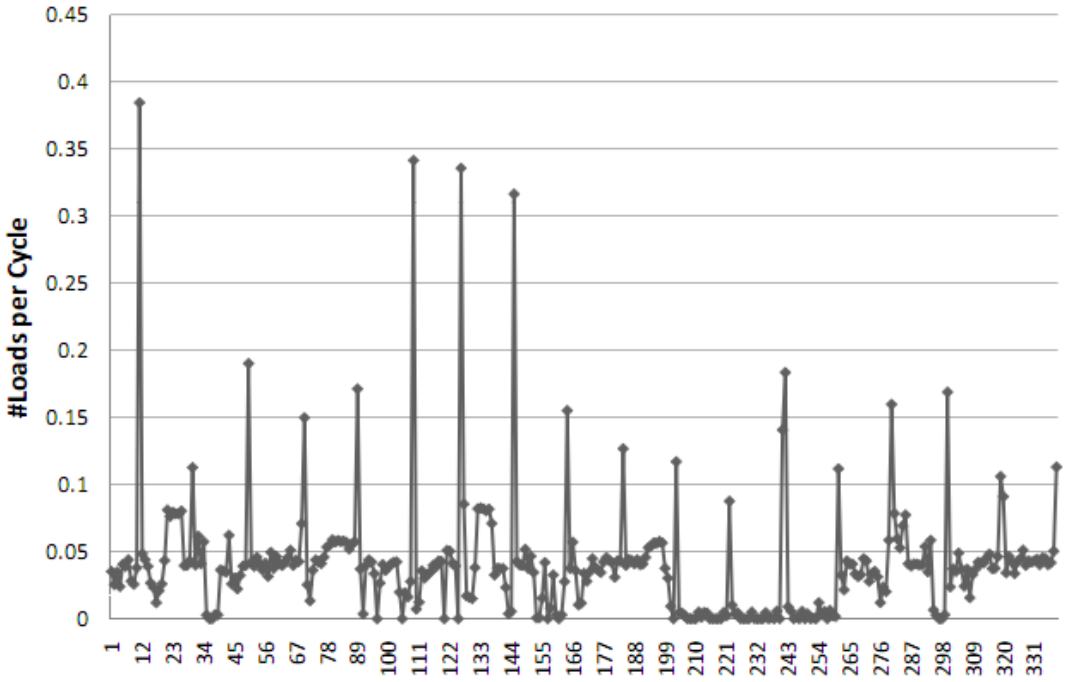


Figure 11: Loads fraction over control intervals.

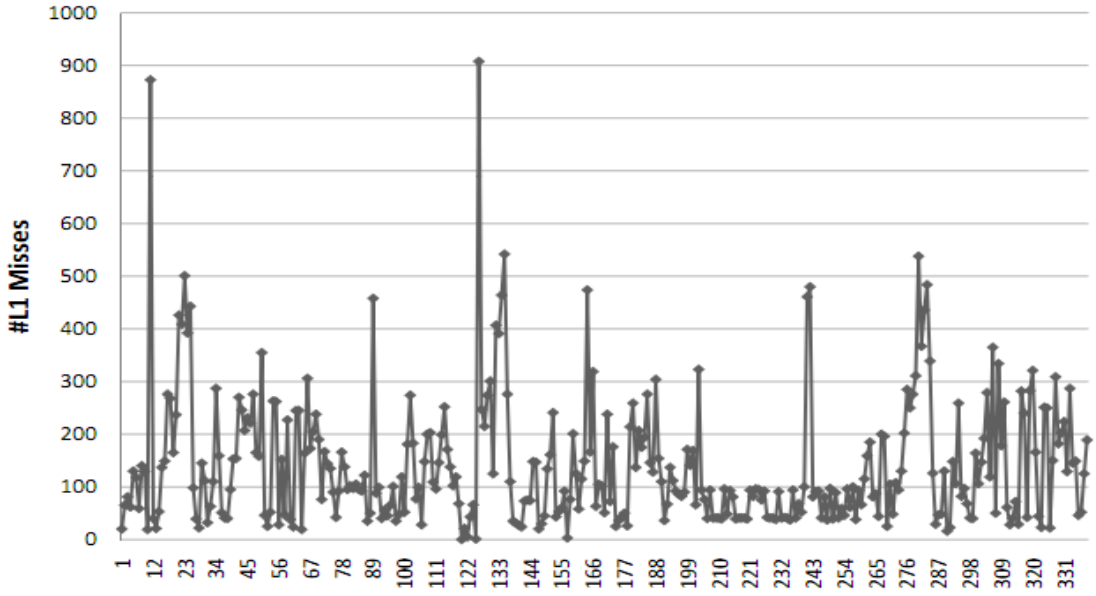


Figure 12: The number of L1 misses over control intervals.

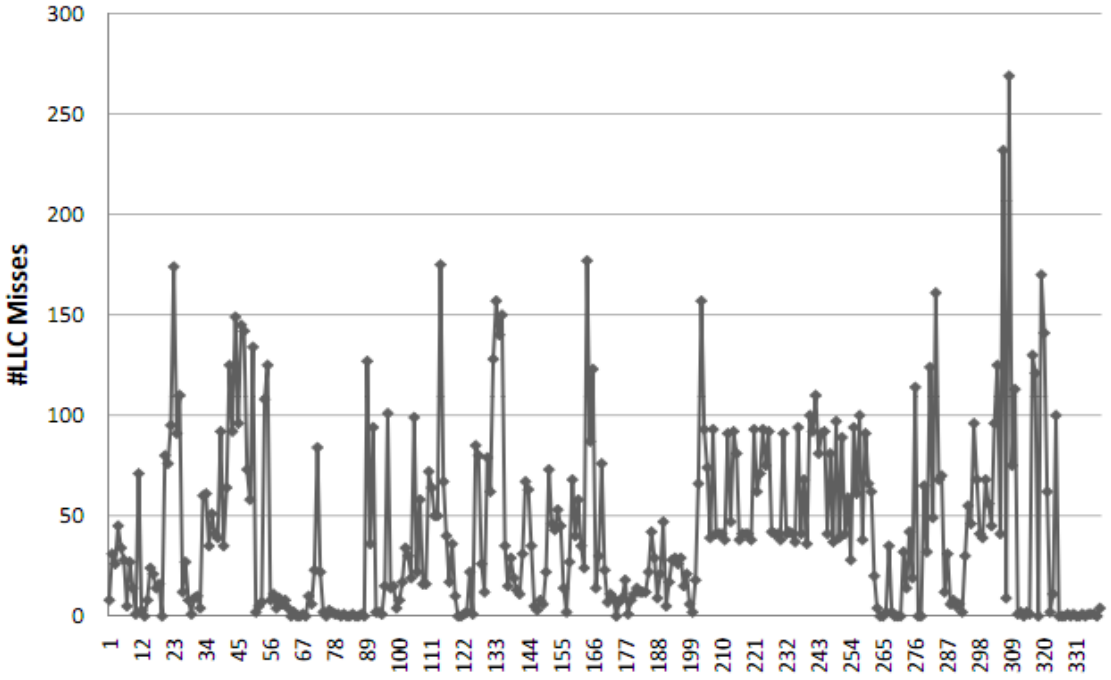


Figure 13: the number of LLC misses over control intervals

Figure 14 below displays the comparison for the different methods as mentioned in previous chapter. From Figure 14, we can see that by using the critical latency proposed in Chen et al.'s work [5], it leads to significantly more power saving compared with using AMAT alone. Especially for the application Canneal, it is very oblivious that the power saving is dramatically increased. Additionally, with the use of model-assist control policy, the power dissipation can be further reduced significantly. It provides about 50% power reduction over Chen et al.'s work [6].

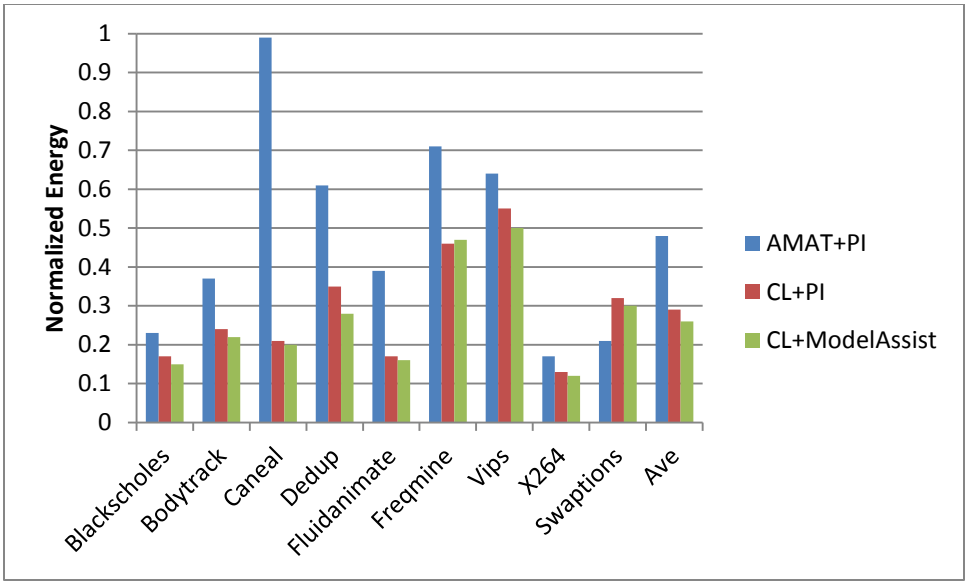


Figure 14: Normalized energy for PARSEC benchmarks

Figure 15 below shows the comparison results of the performance impact under different methods corresponding to the Figure 14. For the proposed approaches, the performance degradation is limited. Dedup is the application which has the largest degradation. If you take a closer look, model assist control actually improves the performance comparing to Xi et al.'s work [6] . In general, the average performance degradation for the proposed model assist control is around only 5%.

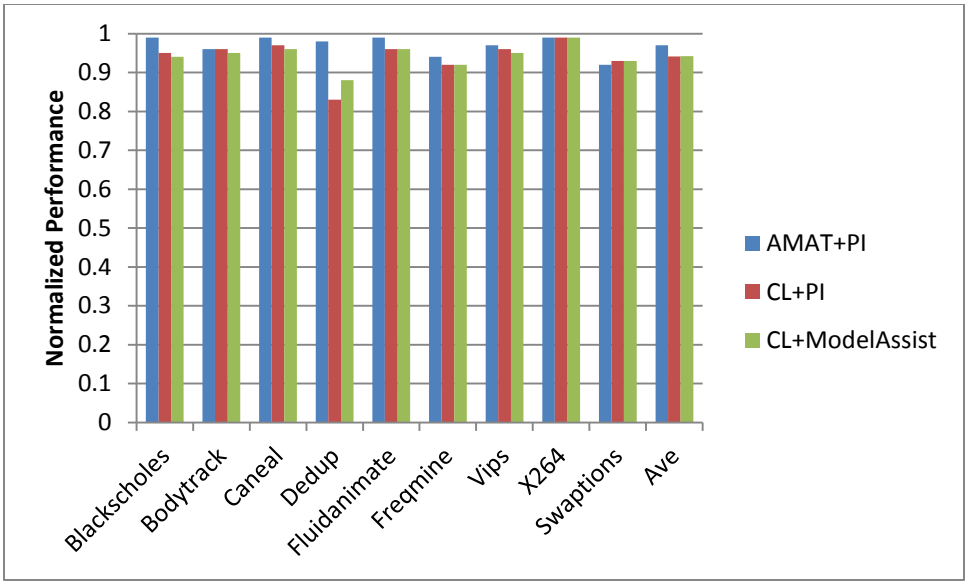


Figure 15: Normalized system performance for PARSEC benchmarks.

Table 2 gives the normalized energy-delay product from all methods. The progressive improvement from the right choice of metric and the benefit brought by proposed new control policy can be obliviously seen through the table. Compared to AMAT+PI [6], our best method reduces the energy-delay product by nearly 50%.

Method	Energy × Delay
Baseline	1.0
AMAT+PI	0.5
CL+PI	0.31
CL+ModelAssist	0.28

Table 2: Normalized energy-delay product on average for all PARSEC cases.

Table 3 gives the normalized energy and performance under different integral bound inside PI controller for preventing wind-up problem. The progressive improvement for the energy saving, and the progressive worse performance loss can be obliviously seen through the table. By comparing these results, we find that a desired energy-performance tradeoff can be obtained when the bound equals 2.5. In that case, it gives average performance degradation for the proposed model assist control around only 5% while achieved a large amount of power saving.

	energy	performance
Base	1	1
Th = 1	0.39	0.97
Th = 1.5	0.33	0.96
Th = 2	0.27	0.95
Th = 2.5	0.26	0.94
Th = 3	0.23	0.92
Th = 5	0.216	0.916
Th = 8	0.212	0.910
Th = 11	0.208	0.912
Th = infinite	0.17	0.88

Table 3: Normalized energy and performance on average for all PARSEC cases under different integral bound

9. CONCLUSIONS

In this thesis research work, we performed DVFS for shared resources (NoC/LLC) in chip multiprocessor systems. The stability of the whole DVFS control system is proved theoretically. The proposed techniques are evaluated on public architecture benchmarks with full-system simulations. Furthermore, the wind up problem in control system is also investigated from both theory and full-system simulations to verify the effectiveness of our anti wind up techniques. In summary, the results show quite large energy savings and improvement over recent previous work.

REFERENCES

[1] Bienia, C., Kumar, S., Singh, J.P., and Li, K. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In Parallel Architectures and Compilation Techniques. 2008.

[2] Binkert, N., Beckmann, B. Black, G., Reinhardt, S.K., et al. The gem5 simulator. ACM Computer Architecture News, 39(2):1-7, May 2011.

[3] Bogdan, P., Marculescu, R., Jain, S. and Gavila, R.T. An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly variable workloads. In Networks-on-Chip Symposium, pages 43-50. 2012.

[4] Chen, X., Physical planning and uncore power management for multi-core processors. Dissertation, Texas A&M University. 2013.

[5] Chen, X., Xu, Z., Kim, H., Gratz, P., et al. Dynamic voltage and frequency scaling for shared resources in multicore processor designs. In Design Automation Conference. 2013.

[6] Chen, X., Xu, Z., Kim, H., Gratz, P., et al. In-network monitoring and control policy for dvfs of cmp networks-on-chip and last level caches. In Networks-on-Chip Symposium, pages 43-50. 2012.

[7] Flautner, K., Kim, N.S., Martin, S., Blaauw, D., et al. Drowsy caches: simple techniques for reducing leakage power. In International Symposium on Computers Architecture, pages 148-157. 2002.

[8] Guang, L., Nigussie, E., Koskinen, L., and Tenhunen, H. Autonomous DVFS on supply islands for energy constrained NoC communication. Lecture Notes in Computer Science: Architecture of Computing Systems, 5455/2009:183-194, 2009.

[9] Kahng, A.B., Li, B., Peh, L.S., and Samadi, K. ORION 2.0: a power-area simulator for interconnection networks. IEEE Transactions On Very Large Scale Integration Systems, 20(1):191-196, January 2012.

[10] Konstantakopoulos, T., Eastep, J., Psota, J., and Agarwal, A. Energy scalability of on-chip interconnection networks in multicore architetures. Technical report, MIT Computer Science and Artificial Intelligence Laboratory, November 2007.

[11] Kowaliski, C. Gelsinger reveals details of Nehalem, Larrabee, Dunnington, 2008.

[12] Kumar, R. and Hinton, G. A family of 45nm IA processors. In International S olid-State Circuits Conference, pages 58-59. 2009.

[13] Liang, G. and Jantsch, A. Adaptive power management for the on-chip communication network. In Proeedings of the Euromicro Conference on Digital System Design. 2006.

[14] Mishra, A.K., Das, R., Eachempati, S., Iyer, R., et al. A case for dynamic frequency tuning in on-chip networks. In International Symposium on Microarchitecture, pages 292-303. 2009.

[15] Muralimanohar, N., Balasubramonian, R., and Jouppi, N.P. CACTI 6.0: a tool to model large caches. Technical report, HP Laboratories, 2009.

[16] Ogras, U.Y., Marculescu, R., and Marculescu, D. Variation-adaptive feedback control for networks-on-chip with multiple clock domains. In Design Automation Conference, pages 614-619. 2008.

[17] Rahimi, A., Salehi, M.E., Mohahmadi, S., and Fakhaie, S.M. Low-energy GALS NoC with FIFO-monitoring dynamic voltage scaling. Microelectronics Jounal, 42(6):889-896, June 2011.

[18] Semaeraro, G., Albonesi, D.H., Dropsho, S.G., Magklis, G., Dwarkadas, S., and Scott, M.L.. Dynamic frequency and voltage control for a multiple clock domain microarchitecture. In International Symposium on Microarchitecture, pages 356-367. 2002.

[19] Shang, L., Peh, L., and Jha, N.K. Power-efficient interconnection networks: dynamic voltage scaling with links. IEEE Computer Architecture Letters, 1(1), 2002.

[20] Son, S.W., Malkowski, K., Chen, G., Kandemir, M., et al. Integrated link/CPU voltage scaling for reducing energy consumption of parallel sparse matrix applications. In International Parallel & Distributed Processing Symposium. 2006.

[21] Wang, H., Peh, L.S., and Malik,S. Power-driven design of router microarchitectures in on-chip networks. In International Symposium on Microarchitecture, pages 105-116. 2003.