Gain Scheduled Control for Nonlinear Power Management in CMPs

Bryan Donyanavard[1], Amir M. Rahmani[1,2], Tiago Mück[1], Kasra Moazemmi[1], and Nikil Dutt[1]
[1]Department of Computer Science, University of California, Irvine, USA
[2]Institute for Computer Technology, TU Wien, Vienna, Austria
e-mail: {bdonyana, amirr1, tmuck, moazzemi, dutt}@uci.edu

***Abstract*—Dynamic voltage and frequency scaling (DVFS) is a well-established technique for power management of thermal- or energy-sensitive chip multiprocessors (CMPs). In this context, linear control theoretic solutions have been successfully implemented to control the voltage-frequency knobs. However, modern CMPs with a large range of operating frequencies and multiple voltage levels display nonlinear behavior in the relationship between frequency and power. State-of-the-art linear controllers therefore under-optimize DVFS operation. We propose a Gain Scheduled Controller (GSC) for nonlinear runtime power management of CMPs that simplifies the controller implementation of systems with varying dynamic properties by utilizing an adaptive control theoretic approach in conjunction with static linear controllers. Our design improves the accuracy of the controller over a static linear controller with minimal overhead. We implement our approach on an Exynos platform containing ARM's big.LITTLE-based heterogeneous multi-processor (HMP) and demonstrate that the system's response to changes in target power is improved by 2× while operating up to 12% more efficiently for tracking accuracy.**

***Index Terms*—DVFS; Non-linear System Dynamics; Gain Scheduling; Control Theory; CMPs; Self-Adaptive Power Management**

I. Introduction

Dynamic voltage/frequency scaling (DVFS) has been established as an effective technique to improve the power-efficiency of chip-multiprocessors (CMPs) [1]. In this context, numerous closed-loop control-theoretic solutions for chip power management [2]–[7] have been proposed. These solutions employ *linear control* techniques to limit the power consumption by controlling the CMP operating frequency. However, the relationship between operating frequency and power is often *nonlinear*. Figure 1 illustrates this by showing total power consumed by a 4-core ARM A15 cluster executing a CPU-intensive workload through its entire frequency range (200MHz–2GHz), along with the total power consumed by a 4-core ARM A7 cluster through its frequency range (200MHz–1400MHz). While the A7 cluster frequency-power relationship is almost linear, the A15 cluster's larger frequency range (and more voltage levels) results in a nonlinear relationship. Using a linear model to estimate the behavior of such a system leads to inaccuracies. Inaccurate models result in inefficient controllers, which defeats the very purpose of using control theoretic techniques for power management.

Ideally, control-theoretic solutions should provide formal guarantees, be simple enough for runtime implementation, and handle nonlinear system behavior. Static linear feedback controllers can provide robustness and stability guarantees with simple implementations, while adaptive controllers modify the control law at runtime to adapt to the discrepancies between the expected and the actual system behavior. However, modifying the controller at runtime is a costly operation that also invalidates the formal guarantees provided at design time.

In this paper we propose a novel nonlinear DVFS power management approach using a well-established and lightweight adaptive control theoretic technique called Gain Scheduling,

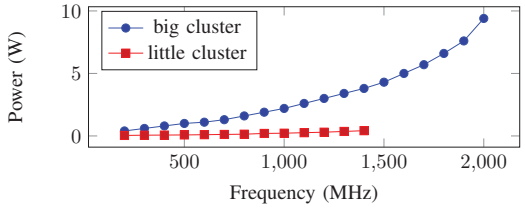


Fig. 1: Cluster power through frequency range.

which to the best of our knowledge has never been exploited previously for on-chip resource/power management. We describe the methodology for integrating multiple linear models within a single controller implementation in order to estimate nonlinear behavior of DVFS for CMPs. We make the following contributions:

- We identify sub-ranges of frequencies that display a linear relationship with power (i.e. *operating regions*), decomposing the entire nonlinear operating region into linear sub-regions, and design static linear feedback controllers for each operating sub-region.
- We design a Gain Scheduled Controller (GSC) for DVFS by adaptively selecting the most appropriate static controller for the current linear sub-region.
- We implement the GSC in a commodity operating system (Linux) and evaluate it on a real platform (ARM CMP), showing that our dynamic controller is up to 12% more efficient in terms of tracking accuracy, and over 50% more responsive than state-of-the-art linear control designs.

II. Background and Motivation

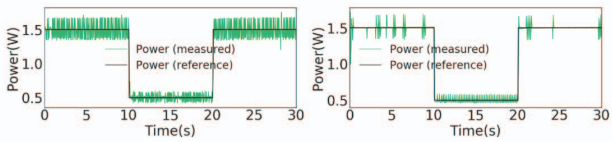
Discrete-time control techniques are the most appropriate to implement control of computer systems. The proportional-integral-derivative (PID) controller is a simple and flexible classical feedback controller that computes control input $u(t)$ based on the error $e(t)$ between the measured output and reference output:

$$u(k) = K_p e(k) + K_i \sum_{0}^{k} e(k)\Delta t + K_d \frac{\Delta e(k)}{\Delta t} \qquad (1)$$

K_p, K_i, and K_d are control parameters for the proportional, integral, and derivative gains respectively.

PI controllers[1] have been successfully used to manage DVFS of CMPs [4]–[7], [9]. Mishra et al. [4] propose the use of PID controllers for VF islands. The authors model power consumption based on the assumption that the difference relationship between power consumption in successive intervals can be approximated linearly as a function of frequency, which only holds for limited range. Similarly, Hoffman et al. [3] propose a feedback control technique for power management that includes DVFS, and their transfer function assumes a linear

[1]Due to the significant stochastic component of computer systems, PI controllers are preferred over PID controllers [8].



(a) Full range SISO controller (1) (b) Sub-range SISO controller (2)
Fig. 2: Time plots of two DVFS controllers tracking a dynamic power reference.

relationship between power and frequency. However, Figure 1 shows that $f \rightarrow P$ becomes nonlinear at higher frequencies. Inaccuracies in linear estimation of nonlinear systems can negatively impact the steady-state error and transient response of the controller. Take for example a system operating under a power budget, or experiencing a thermal emergency – an inefficient DVFS controller could lead to wasted power or even unnecessary operation at an unsafe frequency.

Consider a DVFS controller for a 4-core CMP with a single frequency domain. The first steps in designing a controller are defining the system and identifying the model. The power consumption of our CMP is not linear through the range of operating frequencies supported (200MHz–2GHz), which makes it challenging to model the entire range with a single linear estimation. However, we can divide the measured output (power) for the entire range of frequencies into multiple *operating regions* that exhibit linear behavior. In this example, we identify a model for two different systems: (1) the CMP's behavior through all operating frequencies; (2) the CMP's behavior through a sub-range of the operating frequencies. This specific operating region spans the frequency sub-range of 200MHz–1200MHz. Using these models, we can generate two different $f \rightarrow P$ SISO PI controllers, and compare them using measured SASO analysis [8], focusing on *Accuracy* and *Settling time*. Figure 2 displays each controller's ability to track a dynamic power reference over time for our CMP.

Accuracy is defined by the steady-state error between the measured output and reference input. We calculate the steady-state error as the mean squared error (MSE) between the measured power and reference power. Both controllers are able to track within 1% of the target power. However, the MSE of Controller 2 is 0.003, while that of Controller 1 is 0.013 – an order of magnitude larger. This byproduct of model inaccuracy translates into wasted power and undesirable operating frequency, as well as unnecessary changes in the frequency control input (i.e., increased control effort cost).

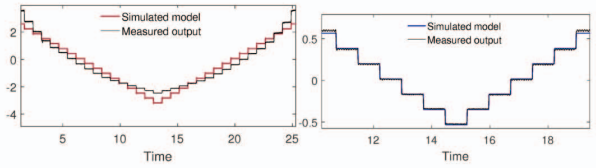
Settling time is the time it takes to reach sufficiently close to the steady-state value after the reference values are set. The settling time of Controller 2 is $40ms$ on average, while Controller 1 is more than double on average at $100ms$. Because our actuation periods are $50ms$, this means that our sub-range controller often reaches steady state on its first actuation while the full range controller requires multiple actuation periods to respond to a change in reference.

Identifying operating regions at design time allows us to switch system models at runtime, improving the effectiveness of static controllers.

III. Designing Gain Scheduled Controller (GSC) for Power Management

In this section, we outline our process for designing gain scheduled nonlinear controllers for a CMP.[2] As a demonstrative case study, we target the ODROID-XU3 platform [12] which

[2]Details to confirm and formalize popular notions regarding gain scheduled design can be found in [10], [11].



(a) Power for full frequency range. (b) Power for 200–800 MHz.
Fig. 3: Modeled and observed behavior of nonlinear full-range system (a) vs. linear operating region (b).

contains an ARM big.LITTLE based Exynos 5422 Octa-core SoC that has heterogeneous multi-processing (HMP) cores. The Exynos platform contains an HMP with two 4-core clusters: the *big* cluster provides high-performance out-of-order cores, while the *little* cluster provides low-power in-order cores. For the purpose of our study, we disable the little cluster (due to its linear behavior) and use only the big cores to emulate a uniform nonlinear CMP[3].

A. Defining and Modeling Linear Subsystems

Selecting the control input and measured output of a DVFS controller is straightforward. Frequency is the knob available to the user in software, and power is the metric of interest. On our Exynos CMP, the operating frequency of cores is set at the cluster level, and power sensors measure power at the cluster level. A SISO controller is a natural solution, with the entire CMP composing the system under control.

For system identification we generate test waveforms from applications and use statistical black-box methods based on System Identification Theory [13], [14] for isolating the deterministic and stochastic components of the system to build the model.

Figure 3a shows a comparison of a simulated model output vs. the measured output over the entire frequency range of our CMP. It is evident that there are ranges for which the estimated behavior differs from that of the actual system behavior. We know that voltage has a nonlinear effect on dynamic power ($P = CV^2f$). The nonlinear relationship between frequency and voltage pairs through the range of operating frequencies amplifies this effect (Table I). Table I lists all valid VF pairs for the CMP, in which there are only four different voltage levels. Figure 3b shows the measured vs. modeled output when the system is defined by a single operating region grouped by frequencies that operate at the same voltage level.

Region	Frequency Range (MHz)	Voltage (V)
1	1600 – 2000	1.25
2	1300 – 1500	1.10
3	900 – 1200	1.00
4	200 – 800	0.90

TABLE I: VF Pairs for ARM A15 in Exynos 5422.

B. Generating Linear Controllers

We generate a PI controller separately for each operating region using the system models and MATLAB's Control System toolbox. This is a straightforward process for a simple off-the-shelf PI controller.

In the next step, the designed controller is evaluated against disturbance and uncertainties in order to ensure it remains stable at a defined confidence level. Unaccounted elements, modeling limitations, and environmental effects are estimated as model uncertainty in order to check the disturbance rejection of the controller. In our case, we can confirm our controller is robust enough to reject the disturbance from workload variation.

[3]We refer to this as the Exynos CMP or CMP throughout.

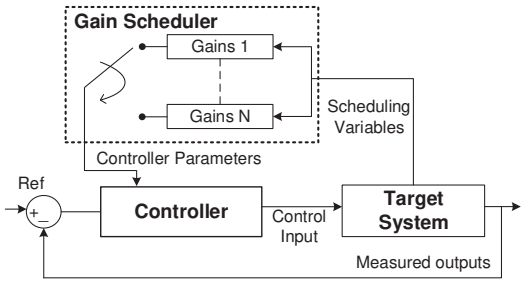


Fig. 4: Block diagram of GSC.

Each controller we design for an operating region is defined by its control parameters K_P and K_I which are stored (in memory) in the gain scheduler (Figure 4). In the gain scheduler, we incorporate logic to determine which gains to provide the controller when invoked.

C. Implementing Gain Scheduling

The gain scheduler enables us to adapt to nonlinear behavior (Figure 4) by combining multiple linear controllers. It stores predefined controller gains and is responsible for providing the most appropriate gains based on the operating region in which the system currently resides each time the controller is invoked.

Algorithm 1 Gain Scheduler Implementation

Input: f: frequency, scheduling variable
Outputs: K_{P_n}, K_{I_n}, $offset_n$: updated controller parameters;
Variables: ref_{prev}, ref_{next}: power reference values for previous and next control periods;
Constants: $Region[N]$: operating regions, defined by mutually exclusive range of frequencies; $K_P[N]$, $K_I[N]$, $offset[N]$: stored controller parameters for each operating region; K_{P_G}, K_{I_G}, $offset_G$: controller parameters for full-range linear controller;

```
1: if ref_next != ref_prev then
2:     K_Pn = K_PG
3:     K_In = K_IG
4:     offset_n = offset_G
5:     return
6: else
7:     for i = 1 to N do
8:         if Region[i].contains(f) then
9:             K_Pn = K_P[i]
10:            K_In = K_I[i]
11:            offset_n = offset[i]
12:            return
13:        end if
14:    end for
15: end if
```

The scheduling variable is the variable used to define operating regions. For our controller, the scheduling variable is frequency as it is simpler to implement in software and has a direct VF mapping (Table I). Our gain scheduler implements lightweight logic that determines the set of gains based on the system's operating frequency (scheduling variable). Algorithm 1 shows the logic implemented in our gain scheduler with N operating regions where f is the scheduling variable and K_P and K_I are the controller parameters. In addition to the K_P and K_I controller parameters, there is also an $offset$. The $offset$ is the mean actuation value for the operating region, and is necessary for providing the control input for the next control period. Algorithm 1 accounts for the transitions between operating regions (lines 1-6) by applying a full-range linear controller. This method is utilized as the sets of gains for a particular operating region perform poorly outside of that region.

	Ctrl 1	Ctrl 2.1	Ctrl 2.2	Ctrl 2.3
Freq. Range	200 – 1800	1300 – 1800	900 – 1200	200 – 800
Stable	✓	✓	✓	✓
Accuracy (MSE)	0.1748	0.03089	0.0005382	0.0003701

TABLE II: Accuracy of the full- (Ctrl 1) and sub-range (Ctrl 2.x) controllers.

IV. Experiments

Our goal is to evaluate our nonlinear GSC with respect to the state-of-the-art linear controller in terms of both theoretical and observed ability to track power goals on a CMP. Our evaluation is done using the Exynos CMP running Ubuntu Linux[4]. We consider a typical mobile scenario in which one or more multi-threaded applications execute concurrently across the CMP.

Controller Configurations: We designed two DVFS controllers for power management of the CMP: 1) **a linear controller** that estimates the transfer function similarly to [3], [4]; and our proposed 2) **GSC**. The GSC contains three operating regions (Table II). We combine the two smallest adjacent Regions, 1 and 2 (Table I), to create Controller 2.1. Controllers are provided a single power reference for the whole system. The control input is frequency, and the measured output is power, applied to the entire CMP.

Implementation: The controller is implemented as a Linux userspace process that executes in parallel with the applications. Power is calculated using the on-board current and voltage sensors present on the ODROID board. Power measurements and controller invocation are performed periodically every $50ms$.

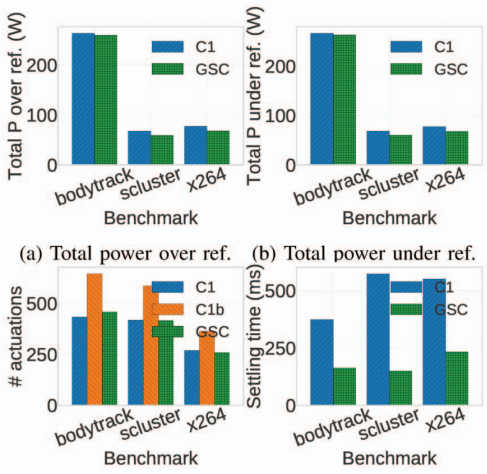
Workloads: We developed a custom micro-benchmark used for system identification. The micro-benchmark consists of a sequence of independent multiply-accumulate operations yielding varied instruction-level parallelism. This allows us to model a wide range of behavior in system outputs given changes in the controllable inputs. We test our controllers using three PARSEC benchmarks: `bodytrack`, `streamcluster`, and `x264`. For each case, we execute one multithreaded application instance of the benchmark with four threads, resulting in a fully-loaded CMP. We empirically select three references that we alternate between during execution. ref_1 is 3.5W, the highest reference and a reasonable power envelope for a mobile SoC. This represents a high-performance mode that maximizes performance under a power budget. ref_2 is 0.5W, the lowest reference and represents a reduced budget in response to a thermal event. ref_3 is 1.5W, a middling reference that could represent the result of an optimizer that maximizes energy efficiency. These references are not necessarily trackable for all workloads, but should span at least three different operating regions for each workload. For each case, the applications run for a total of $65s$. After the first $5s$ (warm-up period) the controllers are set to ref_1 for $20s$, then changed to ref_2 for $20s$, and to ref_3 for the remaining $20s$.

A. Controller Design Evaluation

We used a first-order system, with a target crossover frequency of 0.32. This resulted in a simple controller providing the fastest settling time with no overshoot. Models are generated with a stability focus and uncertainty guardbands of 30%.

All systems are stable according to Robust Stability Analysis. By design all overshoot values are 0. The settling times of Controllers 2.2 and 2.3 are comparably low at 5 control periods. Controller 2.1 (the most nonlinear operating region) and Controller 1 are slightly higher at 8-9 control periods. The ideal controllers are all very similar in terms of stability, settling

[4]Ubuntu 16.04.2 LTS and Linux kernel 3.10.105



(a) Total power over ref. (b) Total power under ref.

(c) Total actuations (d) Average response time

Fig. 5: Comparison of GSC with Controller 1.

time, and overshoot. The primary difference between them is in terms of accuracy. Controllers 2.1-2.3 achieve an order of magnitude better accuracy than Controller 1 (Table II). This means that the region controllers are equally as responsive as the full-range model in achieving a target value while achieving the value more accurately.

B. Controller Implementation Evaluation

We now evaluate the effectiveness of our nonlinear control approach implemented in software on the Exynos CMP for multithreaded mobile workloads. Traditional SASO control analysis gives us a way to compare the controllers in theory, but the system-level effects of those metrics are not directly relatable. Therefore, we will compare the runtime behavior of the software controllers using a slightly modified set of metrics: power over target, power under target, number of actuations, and response time. These metrics are shown in Figure 5.

The *power over target* is the total amount of measured power exceeding the reference power throughout execution (Fig. 5a). This is the area under the output and above the reference. It represents the amount of power wasted due to inaccuracy, and can also represent unsafe execution above a power cap. Our GSC is able to achieve **12%** less power over target than the linear controller for `x264` and `streamcluster`. `bodytrack` is the most dynamic workload and results in the noisiest power output. In this case the GSC only improves the power over target by 1% compared to the linear controller.

The *power under target* is the total amount of measured power falling short of the reference power throughout execution (Fig. 5b). This is the area under the reference and above the output. A lower value translates to improved performance (i.e. lower is better). Similarly to the power over target, our GSC is able to reduce power under target by **12%** for `x264` and `streamcluster`, and 1% for `bodytrack`.

The *number of actuations* is simply a count of how many times the frequency changes throughout execution, and is a measure of overhead (Fig. 5c). The GSC's actuation overhead is lower than the linear controller for `bodytrack`, `streamcluster`, and `x264` by 8%, 1%, and 4% respectively. This is expected, as the controller's resistance to actuation is related to the crossover frequency specified at design time. For the same crossover frequency, the GSC benefits are primarily in the accuracy (power over/under target) and response (settling) time. To illustrate this tradeoff, we performed the same experiments for a full-range linear controller with a target crossover frequency of 0.8 (Controller 1b). We arrived at this value empirically: Controller 1b achieves comparable accuracy to the GSC. However, GSC reduces the actuation overhead by **29%** for all workloads compared to Controller 1b.

The *response time* is the average settling time when the target power changes, indicating the controller's ability to respond quickly to changes (Fig. 5d). Figure 5d shows the average response time for each workload for both controllers. The GSC is able to improve the response time over Controller 1 by more than **50%** in each case. The GSC's overall average response time is $182ms$, which is less than 4 control periods.

The implementation overhead of the GSC w.r.t. the linear controller is negligible: it requires a single execution of Algorithm 1 upon each invocation, and storage for a K_P, K_I, and $offset$ value for each operating region. Although workload disturbance plays a significant role in determining the magnitude in imporovement of a nonlinear GSC over a state-of-the-art linear controller, a clear trend exists, and these advantages would increase with the modeled system's degree of nonlinearity.

V. Conclusion

In this work we present the design of a higher-level nonlinear Gain Scheduled Controller in conjunction with multiple lower-level linear controllers for each linear sub-region, to improve accuracy with minimal overhead. Our design benefits from a simple adaptive control theoretic approach that can facilitate control of non linear systems with dynamic properties. Our evaluations of GSC implemented on a real multi-processor platform using PARSEC benchmarks demonstrated up to 12% power efficiency improvement over a linear controller with minimal overhead and over 50% faster response time. Our GSC approach also demonstrated the ability to tradeoff control overhead with power efficiency, making this a practical approach for deployment on emerging MPSoC platforms.

Acknowledgment

We acknowledge financial support from the following: NSF Grant CCF-1704859, and the Marie Curie Actions of the European Unions H2020 Programme.

References

[1] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *ISLPED*, 2007.
[2] A. M. Rahmani *et al.*, "Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach," in *ISLPED*, 2015.
[3] H. Hoffmann *et al.*, "Dynamic knobs for responsive power-aware computing," in *ASPLOS*, 2011.
[4] A. K. Mishra *et al.*, "Cpm in cmps: Coordinated power management in chip-multiprocessors," in *SC*, 2010.
[5] T. S. Muthukaruppan *et al.*, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *DAC*, 2013.
[6] K. Ma *et al.*, "Scalable power control for many-core architectures running multi-threaded applications," in *ACM SIGARCH Comp. Arc. News*, 2011.
[7] X. Wang *et al.*, "Adaptive power control with online model estimation for chip multiprocessors," *IEEE TPDS*, 2011.
[8] J. L. Hellerstein *et al.*, *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.
[9] Q. Wu *et al.*, "Formal online methods for voltage/frequency control in multiple clock domain microprocessors," in *ACM SIGARCH Comp. Arc. News*, 2004.
[10] J. S. Shamma and M. Athans, "Analysis of gain scheduled control for nonlinear plants," *IEEE Transactions on Automatic Control*, 1990.
[11] D. J. Leith and W. E. Leithead, "Survey of gain-scheduling analysis and design," *International Journal of Control*, 2000.
[12] Hardkernel, "ODROID-XU," Tech. Rep., 2016. [Online]. Available: http://www.hardkernel.com/main/main.php
[13] L. Ljung, *System Identication : Theory for the User*. Prentice Hall PTR, 1999.
[14] ——, "Black-box models from input-output measurements," in *I2MTC*, vol. 1, 2001, pp. 138–146 vol.1.