VR-Scale: Runtime Dynamic Phase Scaling of Processor Voltage Regulators for Improving Power Efficiency

Hadi Asghari-Moghaddam¹, Hamid Reza Ghasemi², Abhishek Arvind Sinkar³, Indrani Paul², and Nam Sung Kim¹

University of Illinois at Urbana-Champaign¹, AMD², Oracle³

Abstract

A voltage regulator (VR) for a processor is one of the most critical platform components. Particularly, a VR is required to support fast, accurate, and fine-grained voltage changes for efficient processor power management. Such requirements, nonetheless, can be relaxed when a processor consumes low power at runtime. Thus, manufacturers begin to offer some knobs so that a processor can adapt VR's operating parameters to cost-effectively satisfy the requirements with high efficiency. In this paper, we first demonstrate that: (1) VR efficiency heavily depends on load current (i.e., current delivered to a processor) and a VR operating parameter (e.g., the number of active phases) at given voltage; (2) a processor running a parallel application mostly consumes small current due to aggressive power management; and (3) when the processor is in active state, all the phases are always activated in the VR. (2) and (3) in turn lead to poor VR efficiency at most runtime. Second, we present VR-Scale that dynamically scales the number of active phases based on the predicted load current for the next interval. Our evaluations based on an Intel processor running emerging parallel applications show that VR-Scale can reduce the total power consumed by a processor and its VR by more than 19% with negligible performance impact.

1. Introduction

One of the most critical components in a platform is a voltage regulator (VR). This is because VRs supply necessary power for various platform components such as processors, chipsets, DRAM modules and storage devices. Overall, the VRs consume 22% of total platform power [1] and they are occupying 63% more platform area than the processor [2]. The key design objective of a VR is to maximize power conversion efficiency, because the power dissipated by the VR is directly proportional to the power consumed by the processor divided by VR efficiency. For example, when supplying 100W for a processor, a VR with 80% efficiency dissipates 25W. At the same time, they must satisfy various operating requirements: delivering stable voltage and large current while supporting fast, accurate, and fine-grained voltage changes for efficient processor power management. To cost-effectively provide such VRs, multi-phase VRs were proposed [3]. A multi-phase VR is comprised of multiple small VRs, each of which operates at a unique phase to share a burden of delivering large current.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DAC '16, June 05-09, 2016, Austin, TX, USA

© 2016 ACM. ISBN 978-1-4503-4236-0/16/06...\$15.00

DOI: <http://dx.doi.org/10.1145/2897937.2898109>

In this paper, we first demonstrate that VR efficiency can significantly vary depending on the amount of current that it delivers to a processor, output voltage, and the number of VR phases in use. For given load current and output voltage, VR efficiency can vary more than 20% depending on how many VR phases are activated. Second, we exhibit that the load current may significantly change over a long time period but it is also very predictable over a short time period, in particular when a processor runs a parallel application. Third, we show that the processor consumes relatively small current for a notable fraction of runtime for running a parallel application. This is because aggressive power saving techniques are applied by the OS. This, nonetheless, leads to very poor VR efficiency since all the VR phases are typically activated unless all the cores in a processor enter low-power idle states. Finally, we present VR-Scale that dynamically scales the number of active VR phases based on the predicted load current.

Some technical briefs from Intel® and HP indicate that some of their products dynamically scale the number of active VR phases (e.g., [4, 5]). However, neither any further technical detail nor benefit analysis is publically available. To our best knowledge, this is the first study that provides the deep technical insights of processors dynamically scaling the number of active phases, analyzes its benefit using a commercial Intel platform, and explores its possible implementation. Note that many commercial VRs complying the most recent Intel VR specification (VR12/IMVP7 [6]) support both circuit- and architecture-level phase scaling techniques (e.g., [7, 8]). The circuit-level technique is also known as auto phase shedding and a VR autonomously changes the number of phases while monitoring its load current. In contrast, the architecture-level technique is directed by the power status indicator (PSI) pins and a VR simply follows a command from the processor and sets the number of phases accordingly. Although these VRs support both techniques, they should be configured to choose either a circuit- or architecture-level technique [7, 8]. Thus, we investigated why the architectural knob to control the number of phases exists regardless of the circuit-level autonomous phase shedding support, surveying many literatures and interviewing industry experts. In summary, we discover that the architecture-level technique can be more cost-effective (i.e., smaller on-chip and/or on-board decoupling capacitors) and more reliable, while the circuit-level technique is useful (at the cost of larger decoupling capacitors) for processors without any phase control mechanism.

The remainder of this paper is organized as follows. Section 2 describes some background. Section 3 analyzes potential VR efficiency improvement by optimally scaling the number of VR phases for given load current and output voltage. Section 4 describes our experimental methodology. Section 5 analyzes runtime current consumption of the processor. Section 6 describes VR-Scale and analyzes power efficiency improvement. Section 7 concludes this paper.

2. Background

Processor power and performance state. Modern commercial processors support C and P states to maximize power efficiency [9]. For example, C0, C1, C3, and C6 states denote operating, halt, sleep, and off states. The deeper the C state is, the lower the power consumption is at the expense of higher performance penalty due to longer wake-up latency. The P0 state indicates the maximum sustainable performance (voltage/frequency (V/F)) state under the thermal design power (TDP) constraint. Similar to C states, the deeper the P state is, the lower the power consumption is, at the expense of lower performance. Besides, as a part of P states, the processors support turbo (or T) states where cores run faster than their P0 state if they operate below power, current, and temperature specification. To support various P and T states, the VR connected to the processor should be able to quickly vary supply voltage.

Platform voltage regulators. Generally, a platform includes a power supply unit (PSU) that converts an AC voltage (e.g., 110V) to various DC voltages (e.g., 12V, 5V, 3.3V, 1.8V, etc.) for platform components such as processor, DRAM, chipset, and hard disk drive (HDD). Since the operating voltages of these components are very diverse, the second-level voltage-down converters (or VRs) are also required on a platform. Such a two-step voltage conversion technique is also commonly used for high efficiency, because directly converting a high AC voltage to low DC voltages is very inefficient.

VRs switch on/off their internal transistors at certain frequency and duty cycle to regulate the output voltage to a desired level. Such VRs desire higher switching frequency (denoted by f_{sw}) for lower output voltage fluctuation and faster responses to the changes of load current. However, simply increasing f_{sw} of a single VR circuit leads to higher power loss. To provide high efficiency at low switching frequency multiple basic VR circuits are placed in parallel between the input and load [10]. Each of N phases is turned on at equally spaced intervals over $1/f_{sw}$, increasing the effective switching frequency by N times without increasing associated switching losses. The number of maximum VR phases depends on the maximum load current required, while typical VRs for high-performance processors use as many as 6-8 phases. Moreover, the number of phases and their efficiency is a function of primarily output voltage, load current, and the number of active phases used to deliver voltage and current for the processor. More parallel phases decrease conduction loss because VR's effective resistance decreases proportional to the number of parallel phases. However, they also increase switching loss because of increasing VR's switching capacitance. Therefore, more active phases offer higher efficiency for higher load current. In contrast, fewer active phases provide higher efficiency

for lighter load current. To maximize efficiency for varying output voltage and load current, most state-of-the-art multi-phase VRs allow us to adjust the number of active phases [11].

3. Efficiency Improvement Opportunity in Multi-phase VRs

3.1 Efficiency vs Load Current and Output Voltage

Figure 1(left) plots the efficiency of a 6-phase VR as a function of load current and the number of active phases at 1.2V output voltage. This VR is optimized to deliver power for an Intel processor that can consume up to 77W at P0 state (i.e., voltage = 1.2V or maximum load current = $77W/1.2V = 65A$). For each 3-tuple of load current (incrementing it up to 65A by 1A), voltage (decrementing it from 1.6V to 0.7V by 10mV), and the number of phases (from 1 to 6), we measure the efficiency of a VR and create an efficiency look-up table after running SPICE based on a model [12] and the following key VR design parameters that impact efficiency: the switching frequency of each phase ($f_{sw} = 300KHz$), inductance ($L = 360nH$) per phase, and parasitic resistance of inductor ($r_L = 0.5m\Omega$) that are derived from a commercial 6-phase VR for an Intel processor [13]; we validate the efficiency for the range of output voltage and load current against a commercial switching VR used for Intel processors [13, 14]. As discussed in Section 2, fewer active phases provide higher efficiency for lighter load current. For example, for 5A load current, using all 6 phases gives only 64% efficiency while using only 1 phase provides 86% efficiency (i.e., 22% higher efficiency than using 6 phases).

Figure 1(right) plots the number of active phases maximizing efficiency as a function of P state (i.e., voltage) and load current, where the maximum load current at each P state decreases as the voltage decreases; our target Intel processor has 20 P states. To obtain the maximum load current for each P state, we scale the maximum load current value at P0 state based on scaling factors obtained by measuring power consumption of various SPEC benchmarks at each P state (cf. Section 4 for the experimental methodology). Figure 1(right) is obtained by searching the number of phases providing the highest efficiency for each pair of load current and P state in the efficiency look-up table constructed for Figure 1(left). Figure 1(right) clearly shows that the optimum number of active phases, which maximizes the VR efficiency, depends on both output voltage and load current. Thus, when all the phases are always activated (or the number of active phases is poorly chosen), we can observe more than 20% efficiency loss. Thus, it is critical to dynamically scale the number of active phases at runtime, because the output voltage and load current of modern processors can significantly vary at runtime.

3.2 Impact on Overall Power Efficiency

While most studies consider the power consumption of only the processor, our study takes the power supplied from the PSU to the VR as total power consumption (P_{TOT}). When the processor consumes P_{PROC} (i.e., power delivered for the processor by the VR), P_{TOT} can be represented as follows:

$$P_{TOT}(n) = \frac{P_{PROC}}{\eta(n)} \quad (1)$$

where n , η , n_{max} , and n_{opt} denote the number of active phases, the VR efficiency, the maximum number of phases, and the optimum number of active phases, respectively. Then the improvement of total power efficiency is expressed by:

$$1 - \frac{P_{TOT}(n_{opt})}{P_{TOT}(n_{max})} = 1 - \frac{\eta(n_{max})}{\eta(n_{opt})} \quad (2)$$

Figure 1: Efficiency as a function of current consumed by a processor and n at P0 state = 1.2V (left). The optimum number of active phases as a function of a P state and load current of a processor (right).

For each pair of P state and load current, we search n_{opt} and apply it to Eq. (2) to compute the improvement. When the load current is less than 5A, optimally scaling n can improve total power efficiency by 25%-63%. The lower the load current and/or the voltage is, the higher the improvement is. We observe that a processor running a single- or multi-threaded application often consumes very small current for a large fraction of runtime. This is because many cores are often idle and thus they often put into very deep P and/or C states; we will demonstrate this in Section 5. For such applications, we can see that optimally scaling n can considerably improve total power efficiency.

3.3 Impact of Runtime Phase Scaling on Performance and Reliability

Note that modern multi-phase VRs began to allow the processor to choose n . The processor can send a command and then n to a VR through PMBus [15]. Considering that PMBus operates at 400KHz and uses a serial interface such as I²C, it takes 60 μ s to send the command and the parameter to the target VR; each adjustment requires the processor to send an 8-bit (VR) address, an 8-bit (phase change) command, and an 8-bit parameter (i.e., n). After the VR PWM controller (de)activates one or more phases, it takes several f_{sw} cycles ($< 30\mu$ s) to stabilize the output current; the exact time depends on the L and C values of the VR design. Thus, a circuit-level autonomous phase scaling technique requires larger on-chip and/or on-board decoupling capacitors to supply the current during the phase change, which increases the cost of the platform but is useful when the processor has no phase control feature. In contrast, an architecture-level phase scaling technique can briefly halt the processor during the phase change, requiring smaller decoupling capacitors. When the processor changes n with the voltage, it does not need to halt according to the VR12/IMVP7 specification [6]. However, just changing the number of active phases without changing the voltage requires the processor to halt for 3.3 μ s [6].

4. Experimental Methodology

For evaluations, we use commercial computer systems based on an Intel[®] i7-3770K processor supporting voltage/frequency 1.52V/3.9GHz~0.72V/1.6GHz, with SMT enabled. We run all the benchmarks on Linux Ubuntu 12.04 after tuning on the "on-demand" and "menu" governors for the P- and C-state management policies; the on-demand governor periodically sets the P state based on the CPU utilization and the menu governor sets the C state based on some heuristics. We use the performance and energy counters in the processor to measure processor power consumption as well as execution time spent at various P and C states (i.e. P- and C-state residencies) every 1ms. In measuring C-state residency, we consider C0 (i.e., non-sleep state including both active and idle states), C1 (halt), C3 (sleep), and C6 (off) states. We measure the frequency of the processor each 1ms interval by reading a special register in the processor and obtain the corresponding VR output voltage (VID) value; we obtain VID values by reading out the MSR IA32_PERF_STATUS value for a dictated frequency value and the measured voltage and power are used to compute the load current (= power/voltage) and the corresponding VR efficiency.

We take PARSEC, DCBench, and SPEC CPU2006 benchmark suites [16, 17, 18] for our evaluation. For all the PARSEC benchmarks, we use the native input set with 8 hardware threads. For SPEC CPU2006, we also evaluate 15 mixes of 2 and 4 co-running benchmarks. The mixes of SPEC CPU2006 benchmarks are based on [19]. For each PARSEC or SPEC CPU2006 benchmark, we run it until completion for multiple times. Finally, we repeatedly execute each mix of SPEC CPU2006 benchmarks such that the execution of

Figure 2: Runtime breakdown by the amount of current consumed by processor (left), P states (center), and C states (right).

one benchmark is overlapped with various starting points of the other one or more co-running benchmarks.

5. Runtime Processor Current Consumption

In this section we demonstrate that a commercial processor running various benchmarks consumes very low current for a substantial fraction of runtime and analyze the root-cause of consuming low current using various measured statistics.

PARSEC. Figure 2(left) plots the runtime breakdown by the amount of current consumed by the processor. In each 1ms measurement interval, the current consumption is obtained by measuring the power consumption every 1ms and dividing it by the voltage associated with the P state of the interval; since a P state changes every 10ms, the voltage during each 1ms measurement interval is constant. When the processor runs a PARSEC benchmark, it consistently consumes substantially lower current than the maximum current for a large fraction of runtime. On average, the processor running a PARSEC benchmark consumes less than 5A and 30A (less than 8% and 48% of the maximum current) for almost 60% and 90% of its total runtime, respectively. Although all four cores are used to execute each parallel benchmark, the power management mechanism aggressively applies deep P and/or C states to maximize power efficiency; for many PARSEC benchmarks using *futex* as a synchronization mechanism, cores spend a considerable fraction of runtime in idle states waiting for synchronizations [20] and they often enter deep C states as shown in Figure 2(right); similar to Figure 2(left), we measure the residency of each P or C state for each core and obtain the average residency of all four cores every 1ms. For example, the cores running *blackscholes*, *bodytrack*, *ferret*, and *vips* spend 60%, 70%, 74%, and 77% of their runtime in C6 state, respectively. Furthermore, they spend 80%, 88%, 62%, and 95% of their runtime in very low voltage/frequency (P16-P20) states, respectively. This leads to relatively low current consumption in most intervals for the PARSEC benchmarks.

DCBench. The processor running a DCBench benchmark consumes somewhat higher current than a PARSEC benchmark, but it still consumes low current for a significant fraction of runtime. We observe that the processor running a DCBench benchmark spends less time in C6 state and considerably more time at high voltage/frequency (T0-T3) states. Our hypothesis is as follow. Many PARSEC benchmarks experience many synchronization events. Consequently,

many cores spend a considerable amount of runtime in idle state. Thus, the power management mechanism often puts these idle cores into deep P and/or C states (e.g., P16-P20 and/or C3-C6). In contrast, many DCBench benchmarks are very data-parallel applications and thus they do not experience many synchronization events. Yet they are often I/O-bound and thus the processor running such a benchmark still spends a notable fraction of their runtime in C6 states and consumes relatively low current. On average, the processor running a DCBench benchmark consumes less than 5A and 30A for almost 30% and 95% of its total runtime, respectively.

SPEC CPU2006. When the processor runs or co-runs one or two SPEC CPU2006 benchmarks at a time, only one or two cores are heavily used while the remaining cores stay in C6 states in most intervals, as depicted by Figure 2(right). Further, the current consumption of the cores is often limited by the maximum operating voltage and/or thermal constraints. Therefore, the current consumption of the processor is relatively low in most intervals. As shown in Figure 2(left), the processor consumes less than 15A (one SPEC CPU2006 benchmark) and 20A (two co-running SPEC CPU2006 benchmarks) for more than 90% of its runtime on average. However, it is expected that the current consumption of the processor co-running four SPEC CPU 2006 benchmarks at a time is fairly high although the measurement is not shown in Figure 2. The processor running four SPEC CPU 2006 benchmarks spends a significant fraction of runtime in C0 state while it also spends most of runtime in high voltage/frequency (T0-T3) states if it is in C0 state. On average, the processor consumes less than 25A, more than 25A but less than 30A, and more than 30A but less than 35A for 25%, 60%, and 16% of their runtime, respectively.

6. VR-Scale

In this section, we first show how the load current changes over time for various benchmarks. Second, based on this observation, we present VR-Scale, a simple technique that dynamically adjusts the number of active phases considering the load current change over time and evaluate its effectiveness in terms of overall power efficiency improvement and performance impact. Third, we analyze the effectiveness of VR-Scale for improving power efficiency of processors supporting per-core voltage/frequency scaling using on-chip VRs.

6.1 Temporal Load Current Change Pattern

One key observation made from this study is that the power consumption of a processor is mainly dominated by the number of cores in C0 state and their P state. This is because fixed power consumption components such as leakage and clocking power dominate the total power consumption of a core in C0 state; we observe that benchmarks showing significant phase changes in performance do

not exhibit as notable phase changes in power consumption in [21] and our experiment using some microbenchmarks.

Figure 3 plots the load current, which is measured every 1ms, of *bodytrack*, *swaptions*, and *canneal* over time. *bodytrack* exhibits a repetitive load current pattern because it repeats for the same computations for each frame. The load current is very low for a while, because only one core is running while the remaining cores are waiting for synchronizations; it was observed that the serial phase of *bodytrack* dominates the execution time. Then the load current considerably increases for a short time period as the synchronizations are resolved and all the cores resume their execution. *facesim*, *ferret*, *vips*, *x264*, and *raytrace* in PARSEC exhibit a similar temporal load current change pattern. All DCBench benchmarks also show a similar temporal load current change pattern except for *hidden markov model (hmm)*, *frequent pattern growth (fpg)*, and *item-based collaborative filtering (ibcf)*. *canneal* shows a few distinct phases of temporal load current change. *blackscholes*, *dedup*, and *streamcluster* in PARSEC also show a similar temporal load current change pattern. For these benchmarks, the number of running threads, which changes over time, dominantly determines current consumption of the processor. *swaptions* show load current notably fluctuating atop a certain offset over time. The fluctuation is incurred by varying execution phases of a thread while the offset is induced by parallelism that always uses all the cores.

Most SPEC CPU2006 benchmarks and their mixes show such a temporal load current change pattern. In fact, SPEC CPU2006 benchmarks show significant phase changes in performance, but they do not exhibit as notable phase changes in power consumption [21] because the fixed power consumption components such as leakage and clocking power dominates the total power consumption of a core in C0 state. This confirms our observation based on the evaluation of SPEC CPU2006 benchmarks. Analyzing the temporal load current change patterns of these three classes of benchmarks, we see that the load current in *ms* time granularity is very predictable and stable although it can significantly change over a longer time period (e.g., hundreds *ms*). Furthermore, the VR-Scale’s prediction of load current for the next interval does not need to be very accurate since VR-Scale attempts to predict the optimum number of active phases (n_{opt}) where a particular n_{opt} value can offer the highest efficiency for a wide range of load current (e.g., from 12A to 23A for $n_{opt} = 2$), as depicted in Figure 1(right).

6.2 Architectural Support and Evaluation

As discussed in Section 2, the PCU, which periodically computes the power consumption of the processor at runtime, is also responsible to control the number of active phases. We present a very simple technique to predict the load current for the next interval. Figure

Figure 3: Load current change over time. The load current and P-state are measured every 1ms.

```

// maximum current for a given number of phases
// to maximize VR efficiency
lut[NUM_PSTATES][NUM_PHASES] = {
{11, 23, 39, 53, 61, 64}, // p_state = 0-3
{12, 24, 40, 53},         // p_state = 4-7
{13, 25, 41, 43},         // p_state = 8-11
{14, 27}};                 // p_state = 12-16

int determine_nopt(int p_state, int current){
    nopt = 0;
    while (lut[p_state/4][nopt++] <= current);
    return nopt;
}

```

Figure 4: VR-Scale runtime algorithm pseudo code running on PCU to determine n_{opt} .

4 describes the pseudo code for the PCU to determine n_{opt} based on a look-up table (LUT) approach.

Runtime algorithm. The LUT is a 2-dimensional array indexed by the P-state value and the number of active phases (n); each entry stores the maximum load current value at which a VR can deliver the highest efficiency for a given P-state value and n . Re-examining Figure 1(right), we see that a range of P states often can share the same row in the LUT since the VR efficiency is a far more strong function of load current than voltage. The number of entries is 16 for the VR used in this study and the LUT entry values can be stored in the platform BIOS chip. Once the LUT is programmed for a given VR, the search algorithm takes at most n_{max} look-ups to determine n_{opt} (i.e., 6 in this study).

Accuracy. Figure 5(top) shows how accurately n_{opt} can be determined by two techniques: VR-Scale and the algorithm based on the current P state of the processor [22], compared to the oracle method. In particular, VR-Scale correctly determines n_{opt} for most SPEC CPU2006 benchmarks for nearly 100% of intervals, because the current consumption is very stable throughout execution, as discussed in Section 6.1. Although VR-Scale incorrectly determines n_{opt} , the chosen n is very close to n_{opt} (e.g., picking $n = 2$ for $n_{opt} = 1$); on average, the difference between n and n_{opt} is less than 0.5 phase for all the benchmarks. In such a case, the negative impact on the power efficiency improvement is very small; the VR efficiency disparity between two neighboring n values is small when the load current slightly exceeds the upper limit for which a particular n value can provide the highest efficiency. In contrast, the algorithm based on

the P state often picks n highly deviating from n_{opt} (e.g., picking $n = 6$ for $n_{opt} = 1$) since many benchmarks consume low current although running at high voltage/frequency P states; on average, the difference between n_{opt} and n at each interval is 2.5, 3.5, and 4.8 phases for PARSEC, DCBench, and SPEC CPU2006, respectively. Consequently, the algorithm based on the P state leads to notably worse power efficiency improvement than VR-Scale for many applications (in particular for running SPEC CPU2006 benchmarks).

Power efficiency improvement. Figure 5(bottom) plots the total power efficiency improvement using VR-Scale and the algorithm based on the P state, relative to a naive approach that always uses all 6 phases. We calculate the overall efficiency improvement of each benchmark as follows:

$$1 - \frac{\sum_{i=1}^N P_{TOT}(n[i])[i]}{\sum_{i=1}^N P_{TOT}(n_{max})[i]} \quad (3)$$

where $n[i]$ denotes n determined by VR-Scale, the algorithm based on the P state [22], or the oracle algorithm at interval i ; $P_{TOT}(n[i])[i]$ ($= PPROC[i]/(\eta(n[i]))$) denotes the sum of processor and VR power consumption (cf. Eq. (1) and (2)) for given $n[i]$ at interval i ; $PPROC[i]$ is the processor power consumption measured at interval i ; and N is the total number of 1ms intervals for a given benchmark. We measure $PPROC[i]$ and voltage every 1ms to calculate the corresponding load current at interval i . Then we determine $n[i]$ based on a given algorithm after looking up the efficiency look-up table for the given pair of the load current and voltage. The oracle algorithm can determine n_{opt} at each 1ms interval.

On average, VR-Scale can improve the power efficiency (defined by Eq. (3)) by 23% and 14% for running two parallel benchmark suites, PARSEC and DCBench (i.e., average power reduction of 1.4~3.8W and 1.1~3.4W for average processor power consumption of 3~34W and 4~32W), respectively. For running one and two SPEC CPU2006 benchmarks at a time, VR-Scale improves the power efficiency by 12% and 7% (i.e., average power reduction of 1.9~2.9W and 1.6~1.8W for the average processor consumption of 10~17W and 21~25W), respectively. Finally, we showed that the processor co-running four SPEC CPU2006 benchmarks consumes very high current for a considerable fraction of runtime, but VR-

Figure 5: Average of n_{opt} by the oracle method - n by current and p-state-based techniques at each interval (top). Power efficiency improvement of current- and p-state-based techniques relative to the VR always activating n_{max} phases (bottom).

Scale can still improve the power efficiency by 4% on average although it is not plotted in Figure 5. On the other hand, the algorithm based on the P state [22] provides notably lower power efficiency improvement than VR-Scale; it improves the power efficiency by 16% and 8%, both of which are about 4% lower than VR-Scale, for running PARSEC and DCBench, respectively. In particular, the algorithm based on the P state practically does not improve power efficiency at all for running one, two, and four SPEC CPU2006 benchmarks at a time while VR-Scale improves power efficiency by 12%, 7%, and 4%, respectively. This is because all SPEC CPU2006 benchmarks mostly run at the highest voltage/frequency P states while they do not consume maximum allowed current.

Performance overhead and reliability. The VR-Scale runtime algorithm is simple and runs on the PCU. Hence, running this algorithm does not incur any CPU performance overhead. Furthermore, while assuming that the processor stalls before changing the number of active phases, we see that the negative performance impact of changing n at runtime is negligible because (i) the time interval is two orders of magnitude longer than the stall time for significantly changing n at runtime, (ii) such stall events are very infrequent, and (iii) n_{opt} in fact does not change frequently. It was shown that sudden load-current changes in Figure 3 are induced by C-state changes of individual cores and such changes are controlled and anticipated by the PCU. Hence, the PCU can proactively handle such cases by halting the processor until the VR stabilizes before such significant load-current changes.

6.3 Processor Supported by On-Chip VRs

We focused our evaluations on a single off-chip VR that provides current for all of the cores in a processor. However, our method can also be applied to newer processor generations having on-chip VRs. In these processors the on-chip VRs share the same technology as the processor; consequently, the VRs cannot directly accept 12V from the PSU. Therefore, an off-chip VR accepts the 12V and provides lower voltages for the on-chip VRs. Thus, we can apply VR-Scale not only to the off-chip VR but also to on-chip VRs.

7. Conclusion

In this paper, we first demonstrate that: (1) VR efficiency heavily depends on load current (i.e., current consumed by a processor) and a VR operating parameter (e.g., the number of VR phases) at given voltage; (2) a processor running one or more applications may consume large current for some periods but mostly consumes small current due to aggressive power management; and (3) unless all the cores in a processor are in sleep states, all VR phases are activated, leading to poor VR efficiency for small load current. Second, we present VR-Scale, an architecture-level technique that dynamically scales the number of active phases based on the predicted load current for the next interval. VR-Scale only requires to run a simple runtime algorithm using the PCU in commercial processors every 1ms. Our evaluations using Intel® Ivy Bridge processor show that VR-Scale reduces the total power consumption of a processor and its VR by 19% and 25%, respectively, with negligible performance impact for two classes of parallel applications. Third, we show that VR-Scale can offer high power efficiency improvement than the algorithm that determines the optimum number of active phases based on the current P state of the processor; many applications run at the highest voltage/frequency P state while consuming low current because not all the cores are always running. Finally, our study opens a door for the architecture community to explore dynamic controls of other VR knobs such as VR switching frequency and adaptive voltage positioning from the CPU side, enabling more cost-effective VRs than the VR side does as argued by [23].

Acknowledgement

This work was supported in part by NSF (CNS-1557233) and DARPA (HR0011-12-2-0019). Nam Sung Kim has a financial interest in AMD and Samsung Semiconductor.

References

- [1] S. Pawlowski, "Driving Towards Cloud 2015: A technology Vision to Meet the Demands of Cloud Computing Tomorrow," 2011.
- [2] Intel Corporation, "Intel Workstation Board S975XB2 Technical Product Specification," 2006.
- [3] X. Zhou, et al., "Investigation of Candidate VRM Topologies for Future Microprocessors," in *IEEE Applied Power Electronics Conf. and Expo. (APEC)*, 1998.
- [4] Intel Corporation, [Online]. Available: <http://www.intel.com/content/www/us/en/servers/technologies/efficient-power.html>.
- [5] HP, *Power efficiency and power management in HP ProLiant servers*.
- [6] Intel Corporation, *VR12/IMVP7 Pulse Width Modulation (PWM) Specification*, 2009.
- [7] Intersil, "ISL6367 Green Hybrid Digital Dual 6+1 Phase PWM Controller for VR12/IMVP7 Applications With SMBus/PMBus/I2C and AUTO Phase."
- [8] ST, "PM6764, PM6766 VR12.5 digital multiphase controller with PMBus."
- [9] [Online]. Available: <http://www.acpi.info/>.
- [10] X. Zhou, et al., "A Novel Current-sharing Control Technique for Low-voltage High-current Voltage Regulator Module Applications," *IEEE T. on Power Electronics*, vol. 15, no. 6, pp. 1153-1162, Nov 2000.
- [11] ON Semiconductor, "Programmable Multi-Phase Synchronous Buck Converter," 2008.
- [12] J. Lee, et al., "Evaluation of Fully-Integrated Switching Regulators for CMOS Process Technologies," *IEEE T. on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 9, pp. 1017-1027, Sep 2007.
- [13] International Rectifier, [Online]. Available: <http://www.irf.com/technical-info/whitepaper/pswus03vrmdesign.pdf>.
- [14] Texas Instrument, [Online]. Available: <http://www.ti.com/product/TPS51631/technicaldocuments>.
- [15] [Online]. Available: <https://pmbus.org>.
- [16] Princeton University, [Online]. Available: <http://parsec.cs.princeton.edu/>.
- [17] Z. Jia, et al., "Characterizing Data Analysis Workloads in Data Centers," in *IEEE Int. Symp. on Workload Characterization (IISWC)*, 2013.
- [18] Standard Performance Evaluation Corporation, "SPEC CPU2006," 2006.
- [19] A. Jaleel, et al., "Adaptive Insertion Policies for Managing Shared Caches," in *ACM Int. Conf. on Parallel Architectures and Compilation Techniques (PACT)*, 2008.
- [20] J. Demme and S. Sethumadhavan, "Rapid Identification of Architectural Bottlenecks via Precise Event Counting," in *IEEE/ACM Int. Symp. on Computer Architecture (ISCA)*, 2011.
- [21] V. Spiliopoulos, et al., "Power-Sleuth: A Tool for Investigating your Program's Power Behavior," in *Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOT)*, 2012.
- [22] J. Jenne, "Dynamic CPU Voltage Regulator Phase Shedding". U.S.A. Patent US20110320838 A1, Dec 2011.
- [23] D. Freeman, "Digital Power Control Improves Multiphase Performance," *Power Electronics Technology*, pp. 2-3, Dec 2007.