TAP – Token-Based Adaptive Power Gating

Andrew B. Kahng[†‡], Seokhyeong Kang[‡], Tajana Rosing[†‡] and Richard Strong[†] [*]
[†]CSE and [‡]ECE Departments, University of California at San Diego
La Jolla, CA 92093-0404 USA
abk@ucsd.edu, shkang@vlsicad.ucsd.edu, tajana@ucsd.edu, rstrong@eng.ucsd.edu

ABSTRACT

We propose a low-overhead technique, Token-Based Adaptive Power Gating (TAP), to power gate an actively executing out-of-order core during memory accesses. TAP tracks every system memory request, providing a lower-bound estimate for the response time. TAP also tracks the state of every power-gateable core in the system, to provide minimal latency wake-up modes to cores such that voltage noise safety margins are not violated. A power-gating switch that utilizes TAP can deterministically power gate its core with energy savings up to 22.39% and no performance hit.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Microprocessors and microcomputers*; C.0 [**Computer Systems Organization**]: General

Keywords

power gating, core, memory access, voltage noise

1. INTRODUCTION

During every cycle that a core is on, even when stalled, leakage power is consumed via gate leakage, gate-induced drain leakage, junction leakage, and subthreshold leakage. A core may stall quite often if it is intensely accessing the memory subsystem, as every time a thread makes a memory request that misses in the L1 cache, the core is subjected to a variable access latency. This variable latency often translates into a core stall during which no forward thread progress occurs and energy is wasted. For a 32nm out-of-order EV6 core, stall energy can be up to 39.1% of total energy consumption for the Spec2006 benchmarks [12].

Power gating is a technique that drastically reduces leakage power by cutting off the current path from supply to ground through introduction of a transistor switch between them. At one end of the spectrum, functional unit power gating reduces power consumption of unused core functional units [22] with wake-up latencies of several nanoseconds. At the other end, entire cores may be power gated and woken up with latencies of several hundreds of milliseconds to account for saving and restoring all core state from memory [11]. The authors of [9] propose to power gate in-order cores stalled on memory misses. Their programmable power gating switch (PPGS) enables core wake-up latencies as small as 8.06ns in *EV4* cores (with 6.25*M* transistors and 32nm technology). Core state is restored by saving critical sequential cells in retention flip-flops, and retaining SRAM cells with source biasing at a cost of less than 3.5% area overhead. The PPGS is directed by a counter-based controller that power gates the core after a stall is detected for longer than a last-level cache hit. The switch resumes core execution at the predicted memory response time, which dynamically adapts to memory latency. However, without detailed knowledge of all core memory requests, out-of-order execution and hard-to-predict stalls impede use of this power-gating mechanism.

This paper proposes a new system, *Token-based Adaptive Power Gating* (TAP), that deterministically applies power gating during core stalls which are caused by the variable latency of requests to the memory subsystem. TAP achieves this by providing two capabilities. First, TAP informs a core's PPGS about the lower-bound latency of memory requests that miss in a given cache level. The expected latency is sent to each PPGS by enabling the cache controllers to send tokens on a miss that include a lower-bound estimate of the access latency of a next-level memory hit. Second, TAP ensures that each core's PPGS uses a wake-up mode that does not violate supply voltage noise constraints of the system when waking up a core. This is ensured by having each core register its state with a centralized *wake-up controller* (WUC), detailing whether the core is idle or active; the WUC responds with a safe wake-up mode to all registered cores. We determine safe wake-up modes by analyzing every possible combination of core states and locations during core wake-up, and creating an equation that models our findings. Any core's PPGS controller, interfaced with the TAP system, has information regarding a lower bound on the time needed to complete all memory requests and a corresponding safe wake-up mode. Combining these two pieces of information, such a controller may determine when to power gate so as to avoid any performance hit, while still ensuring energy savings.

Our work makes the following contributions:

- A token-based system (TAP) to deterministically and safely manage power-gating constraints with energy savings as high as 22.39% for a wake-up latency of 10.2ns.
- Compared to a previous state-of-the-art technique [9], TAP has 2.58× the average energy savings for an out-of-order core. TAP is also shown to have better energy savings than a practical DVFS policy at 32nm.
- An analytical model that accurately estimates core wake-up modes for an arbitrary multi-core wake-up scenario.
- A new technique that introduces *wake-up stagger* between cores to reduce the minimum feasible core wake-up latency by up to 40.3%.

[*]Authors are listed alphabetically by last name. Principal contributors, to whom correspondence should be addressed: Richard Strong (rstrong@eng.ucsd.edu) and Seokhyeong Kang (shkang@vlsicad.ucsd.edu).

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. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ISLPED'12, July 30–August 1, 2012, Redondo Beach, CA, USA.
Copyright 2012 ACM 978-1-4503-1249-3/12/07 ...$10.00.

2. RELATED WORK

Power gating has been widely studied at both architectural and circuit levels. The following briefly reviews representative works in these two areas. Hu et al. [8] propose power gating as a technique to reduce functional unit leakage power, when applications underutilize their functional units. Specifically, they power gate the floating-point and fixed-point units according to three different predictors, which are respectively ideal, time-based, and branch-misprediction-guided. The best technique, branch-misprediction-guided, is able to put functional units to sleep for up to 40% of total cycles with only 2% performance loss. The authors of [8] also develop equations to estimate the break-even points for power gating an out-of-order superscalar processor. Although they build a power consumption model with precise analysis of virtual supply voltage during power gating, they do not consider the wake-up energy to restore circuit nodes, which will impact energy savings.

Lungu et al. [13] show that in many cases, the predictor of [8] can lead to increased energy consumption. A monitor that controls the use of power gating is introduced to bound the performance and energy penalty for misbehaved applications. Madan et al. [14] extend the idea of Lungu et al. to the core level, and propose a "guard mechanism" that reduces harmful use of power gating.

Power gating technology is also readily visible in leading commercial products. The recent Nehalem architecture employs power gating at the core level to reduce leakage power on idle cores, but 100ms is required to wake-up a core [10, 11]. In today's systems, the OS only power gates cores in the idle loop which ignores potential for core energy savings during long memory accesses.

At the circuit level, the pioneering work of Horiguchi et al. [7] has been followed by many publications on fundamental circuit design issues related to power gating, including switch cell sizing, data retention methods, physical implementation methodologies, and mode-transition noise analysis and reduction. The recent survey of Shin et al. [17] gives an excellent summary of the history and highlights of power gating technique.

Configurable power gating has been introduced in the past to mitigate process variation, reduce ground bounce noise, and minimize wake-up time. Agarwal et al. [3] and Singh et al. [18] examine multiple sleep modes that feature different wake-up overheads and leakage power savings. Use of multiple sleep modes achieves an extra 17% reduction in leakage power compared to a single power gating mode. Also, one of the sleep modes can reduce leakage power by 19% while preserving circuit state. However, these energy savings are based on static traces of bus activity and do not address the runtime problem of predicting when to power gate. In addition, the reported results are likely optimistic since wake-up noise is neglected, and the overhead of implementing low-voltage sleep control signal distribution is not considered.

The closest work to our own is MAPG [9], which proposes a technique to power gate in-order cores during long memory accesses. However, the MAPG work considers neither out-of-order execution nor the benefits of exploiting core location and state information for determining safe wake-up modes. Our present work addresses these issues, shows the importance of stagger to reduce core wake-up latency, and provides a comparison between TAP, MAPG-Counter, and DVFS for out-of-order cores.

3. POWER GATING AND PDN ANALYSIS

This section provides an analysis of our power-gating methodology, and its impact on the power distribution network. Power gating cuts off leakage current paths between supply and ground by using switch transistors (such as high-Vth or long-channel devices). To power gate a circuit, header switches turn off, and leakage current is reduced. While in the power-gated state, all logic gates connected to the virtual supply lose their logical states. Resuming circuit operation incurs a delay that corresponds to charging circuit capacitance, resetting memory elements, and restoring state.

To maximize power gating opportunities, we use a *programmable power gating switch* (PPGS) [9]. The PPGS offers multiple wake-up modes with different wake-up latencies; this enables greater leakage savings when lower-latency wake-ups are feasible. The PPGS works as a two-stage wake-up controller [6]. In the first stage, the signal *enable_few* turns on a subset of header switches to allow I_{limit} charging current until the circuit nodes are nearly charged. In the second stage, the signal *enable_rest* turns on the remaining header switches, resulting in a triangular charging current profile. The PPGS's wake-up time and inrush current are determined by the number of header switches that are turned on by the first-stage *enable_few* wake-up signal. The two key constraints on the choice of wake-up modes are that the voltage noise must be less than 5% on neighboring active cores, and less than 40% on neighboring idle cores [9].

To study wake-up latency and inrush current, we estimate the total charge for core logic and interconnect capacitance using the methodology of [9]. We determine the core area, power, and inrush current limits with McPAT [12]. Core transistor count, capacitance, and Vdd_core are determined from the 2009-2010 *International Technology Roadmap for Semiconductors* (ITRS) [1]. We summarize these values in Table 1.

Estimated Data	32*nm* HP	32*nm* LOP	22*nm* HP	22*nm* LOP
Design Data				
Vdd_core (V)	1.00	0.77	0.93	0.72
core area (mm^2)	15.894	16.185	8.616	8.608
logic area (mm^2)	11.513	11.568	6.531	6.527
C_{core} (F)	30.7E-9	30.8E-9	18.7E-9	18.7E-9
total charge (C)	30.7E-9	23.7E-9	17.4E-9	13.4E-9
core leakage (W)	0.916	0.143	0.572	0.076
I_{active} (A)	2.665	1.690	1.721	1.152
I_{limit} (A)	13.544	10.712	7.834	6.871
Power Gating and Wake-up				
$T_{min-charge}$ (ns)	7.80	5.40	4.40	3.40
wake-up energy (pJ)	15.40E3	9.16E3	8.11E3	4.86E3
# head switches	126,435	105,295	72,188	67,655
leakage in PG state (W)	21.51E-3	2.01E-3	14.62E-3	1.25E-3
leakage reduction in PG	97.65%	98.60%	97.49%	98.37%

Table 1: Estimated data for 32*nm* and 22*nm* EV6 cores.

We use the detailed PDN model from [9] that includes all package parasitics, to enable realistic noise analysis under various wake-up scenarios. Power is delivered from an external voltage regulator module (VRM) through a printed circuit board (PCB), a package ball, package interconnect, microbumps, on-die redistribution layers, the on-chip PDN, and power gating switches. We model the entire power delivery network, including power gating switches, as a simplified RLC circuit as designed in [9]. Package inductance and series resistance from VRM to bumps for a core are lumped as in-series inductance, $L_{pkg-core}$, and resistance, $R_{pkg-core}$. The PDN in the package, which is shared by multiple cores, is represented as a resistance mesh with a branch resistance of R_{shared}. There are three variant models depending on the state of the core — *core in active mode*, *core being woken up*, and *core in sleep mode*. We use the PDN parameter values from Table I of [9].

Exploiting Spatial Information. Previous PPGS work [9] selected a wake-up mode based on the worst-case wake-up time for each *number* of idle cores. The worst-case wake-up time assumption limits the benefit of power gating. A core's minimum wake-up time is constrained by the voltage noise seen by neighboring active cores – in particular, some *critical* active neighbor core, where the voltage noise constraint is first violated. The voltage noise of an active core is mainly affected by adjacent woken-up cores and the latencies (i.e., associated inrush currents) with which they wake up. In other words, we may exploit knowledge of a core's location to

reduce pessimism. We have developed a model that determines the minimum wake-up time based on the number and location of active and woken-up cores. To simplify the model, we assume that all woken-up cores have the same (uniform) wake-up latency.

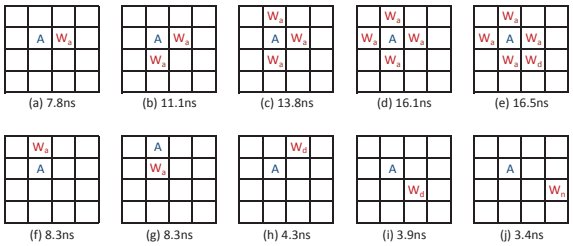

Figure 1: SPICE-calculated minimum wake-up latency for an EV6 16-core CMP with various wake-up scenarios.

Figure 1 shows the minimum wake-up time, according to the location and status of cores, for an example case of an EV6 16-core CMP where A denotes the critical active core, W_a are adjacent woken-up cores, W_d are diagonally adjacent woken-up cores, W_n are non-adjacent woken-up cores, and blank squares are idle or non-critical active cores. The PDN model of [9] is used The wake-up latency increases according to the number of adjacent woken-up cores (Figure 1 (a) - (e)), and is approximately proportional to the square root of the number of adjacent woken-up cores. Woken-up cores in the diagonal (W_d) or non-adjacent positions impact wake-up latency less than adjacent woken-up cores (Figure 1 (f) and (g)). If woken-up cores are located at an edge position (Figure 1 (h)), the minimum wake-up time increases.

From such observations, we have modeled the minimum wake-up latency based on the core status at each location as:

$$T = T_0(w+\beta\cdot x+\gamma\cdot y+\delta\cdot z)^{\alpha} \quad (1)$$

where T_0, α, β, γ and δ are fitting coefficients, w is the number of adjacent woken-up cores, x is the number of diagonal woken-up cores, y is the number of other (non-adjacent) woken-up cores, and z is the number of cores (active core itself or adjacent woken-up cores) which are at edge locations.

We have verified our model with SPICE, and modeled the wake-up time for 4-, 6-, 8-, and 16-core CMPs for all location permutations. Our SPICE simulations measure the voltage drop on active cores to find the minimum wake-up time.

The results in Table 2 show that our model has average error of 2.64%, 1.93%, 2.31% and 1.57% for 4-, 6-, 8-, and 16-core CMP cases, respectively. The parameter values can be adjusted in real IC designs to adapt to process, voltage and temperature variations.

core	coefficient					error	
	T_0	α	β	γ	δ	average (%)	maximum (ns)
4-core	7.9	0.50	0.35	0.15	0.15	2.64	0.37
6-core	7.9	0.50	0.35	0.15	0.13	1.93	1.10
8-core	7.9	0.50	0.30	0.15	0.13	2.31	1.65
16-core	7.9	0.50	0.20	0.10	0.10	1.57	1.40

Table 2: Average and maximum error on the modeled wake-up time for 4, 6, 8, and 16-core cases (EV6, 32*nm* HP).

Sensitivity of Results to PDN Parameter Values. The results of the SPICE simulation have varying degrees of sensitivity to power distribution network (PDN) parameters - the number of bumps, package inductance (L_pkg), package resistance (R_pkg), PDN mesh resistance (R_shared), supply voltage and core capacitance. We have assessed the minimum wake-up time sensitivity to variations in the PDN model. Figure 2 shows the change in the T_0 coefficient when each PDN parameter is scaled from 0.1× to 2× (a 20× range!) with respect to our default values, which are obtained from personal communication with industry experts. Since the actual wake-up latency depends on PDN variations, the T_0 coefficient will be determined by testing the actual packaged chip. Our conclusions regarding energy savings and overheads remain qualitatively the same across the range of PDN parameter values.

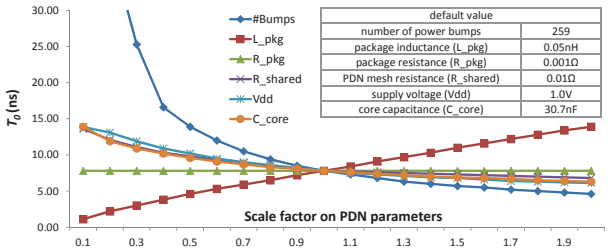

Figure 2: T_0 as a function of PDN parameters.

Benefits of Staggered Wake-up. The above wake-up analysis assumes that all cores wake up simultaneously, which is the worst case. However, wake-up latency is significantly reduced when we *stagger* the wake-up sequence so that two cores wake up at slightly different times (e.g., offset by 1ns). (Staggered wake-up is analogous to multi-stage wake-up control within a core.) We design the wake-up controller to insert stagger between waking cores to reduce wake-up latency. Figure 3 shows minimum wake-up latency for an EV6 16-core CMP when we add stagger between woken-up cores. The minimum wake-up time (y-axis) is reported for the worst case for each number of woken-up cores (x-axis). When stagger is zero, wake-up time increases according to the number of woken-up cores. However, if we avoid simultaneous wake-up, minimum wake-up time reduces greatly. When two, three, and four cores are waking up within an interval of three cycles (0.9ns), we obtain 18.8%, 31.9% and 40.3% wake-up latency reductions, respectively, over simultaneous wake-up. From SPICE results in Figure 3, we can see that the minimum wake-up time does not increase with staggered wake-up when the number of woken-up cores is larger than four. We have modeled the minimum wake-up time with Equation (1) for up to three woken-up cores by changing the parameter α from Table 2. The dotted lines in Figure 3 show the modeled wake-up latency from Equation (1) and its error with respect to SPICE simulation. Our measurements of model accuracy show an average (maximum) error of 2.66% (7.62%), 1.89% (6.61%), 0.93% (3.59%) and 2.51% (3.08%) for the 4-, 6-, 8-, and 16-core CMP cases, respectively.

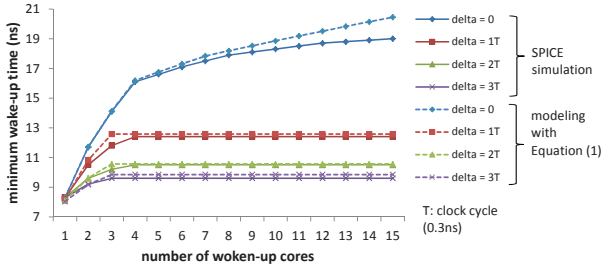

Figure 3: Minimum wake-up latency versus wake-up stagger.

4. SYSTEM DESIGN

We now present our architectural modifications used to control power gating for each core. As noted above, TAP is a system with two purposes, (1) to inform a core's PPGS about the expected latency of memory requests that miss in a given cache level, and (2) to ensure that each core's PPGS uses a wake-up mode that does not violate reliability constraints of the system when waking up a core. Together, these two pieces of information allow each PPGS to power gate its core as soon as a stall on a memory operation is detected, and the expected latency of the memory operation is greater than the break-even point for power gating.

Determining Lower Bound on Stall Operation. We achieve the first goal of informing each PPGS about expected memory latency by modifying the cache controllers to send tokens on cache misses. The tokens include an estimate of the lower-bound access latency for a next-level memory hit, derived from Table 3, and a time stamp of creation. The controllers send the tokens to the PPGS of the core that requested the memory access. Once the PPGS receives the token, it looks at the lower-bound latency to satisfy the request and power gates the core if the core is both stalled and idle long enough to save energy. Should the core receive more than one token for simultaneous memory requests, it will track each expected response separately and schedule the resumption of core execution to satisfy the earliest response. If a token is delayed in the memory subsystem by a controller or queue, the PPGS can compare its arrival time with its generation time stamp and previous tokens, to determine whether the token should be ignored.

Whenever a memory request misses all the way to the memory controller, the response latency experiences a significant amount of variability. This variability is caused by the complexity of DRAM memory [19], which includes bank queues, availability of the data in the row buffer, writing wrong address row buffers, accessing the column in the row buffer, and channel contention between banks. TAP adapts to memory variability by adding a special token. As soon as the last-level cache experiences a miss, a token is sent to the requesting core's PPGS with an estimated completion time of *UNKNOWN*. This is a directive to the PPGS to start power gating its core immediately and to expect one additional token with the ETA of the memory response. Once the memory controller submits the memory access to one of the banks and determines whether the access is a row buffer hit or miss, it sends the second *ETA* token to the core's PPGS with the ETA of the response assuming that there is no memory channel contention. The PPGS schedules the core to wake-up after it receives the second token.

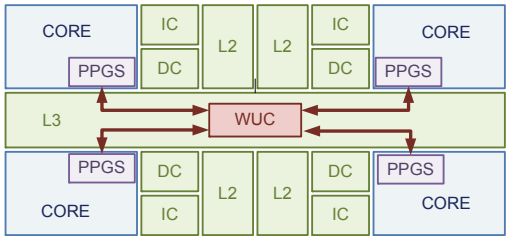

Figure 4: WUC and PPGS integration into a 4-core CMP.

Distributing Safe Wake-Up Modes. To achieve safe and minimal wake-up latencies for a power-gated core, TAP requires each core to register with the wake-up controller (WUC) (see Figure 4), and set whether it is *active* or *idle*. If the PPGS registers with the WUC as *active*, the WUC returns the worst-case lowest latency wake-up mode defined by Equation (1). If the PPGS registers with the WUC as *idle*, the WUC will send the new wake-up mode to the remaining active cores. As soon as the system boots, firmware causes each core's PPGS to send a command packet to the WUC. A core is neither allowed to power gate during a memory stall, nor allowed to wake-up from a power-gated state, until it has registered with the WUC and been assigned a wake-up mode. This prevents any violation of voltage noise constraints.

When a core stalls and power gates, the PPGS may use the worst case wake-up latency it initially received from the WUC and save energy. However, this results in unnecessarily long wake-up latency and reduced energy savings. To enable stagger and more aggressive wake-up modes as a function of whether neighboring cores are active, waking-up, or idle, the PPGS may query the WUC for a lower latency wake-up. If the WUC replies in time, the PPGS may delay its wake-up procedure for the difference between the worst-case and actual wake-up delay defined by Equation (1). Because only adjacent cores have a significant effect on a core's wake-up latency, with appropriate wake-up latency guardband, only adjacent cores must communicate with the WUC, which can be done before the PPGS must start the wake-up sequence. Often, the result is lower core wake-up latency and greater energy savings.

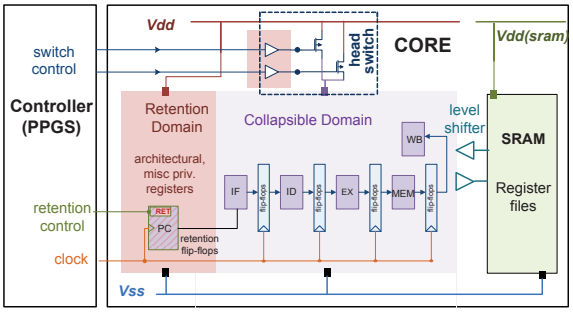

Figure 5: Interface for power gating and data retention.

Retention of Essential Core State. To avoid losing core state that is required for correct and efficient execution, essential sequential and SRAM cells must be retained. We use the technique from [9] which replaces a subset of sequential cells with live-slave retention flip-flops [5] which can be triggered to retain their logical values before a power gating action at a cost of 20% increase in area and power versus a normal flip-flop. Only those sequential cells comprising the architectural registers necessary to refill the pipeline are selected, which results in 3.4% area overhead for the processor. SRAM cells are retained through source biasing [16] in which the supply voltage is reduced to 50% of nominal supply voltage so that SRAM leakage is reduced, but logical state is maintained. This technique allows for saving the contents of L1 caches, TLBs, branch predictor state, physical registers, etc. To provide supply power during power gating, a separate non-collapsible voltage domain provides power to the retention flip-flops and SRAM cells. Thus, as the power is gated from combinational logic and non-essential sequential cells, the separate voltage rail provides power to maintain core state. The overheads from multiple power domains and separate voltage rails already exist for power-gated cores today. Figure 5 shows core design modifications to support power gating and state restoration.

To retain internal data during power gating, additional cycles are required for the power gating and wake-up sequence. These additional cycles account for time to disable/enable the clock, trigger data retention, refill the pipeline, and de-assert/assert the clamps. We model the entire power down and wake-up sequence as in [9].

Figure 6 shows a timing-accurate diagram of a PPGS power gating the core in response to messages from the TAP system. At time 0ns, a memory request occurs that will miss in the cache hierarchy and cause a memory access. The PPGS then receives tokens for the L1, L2 and L3 misses. Just after receiving the L2 Token, the

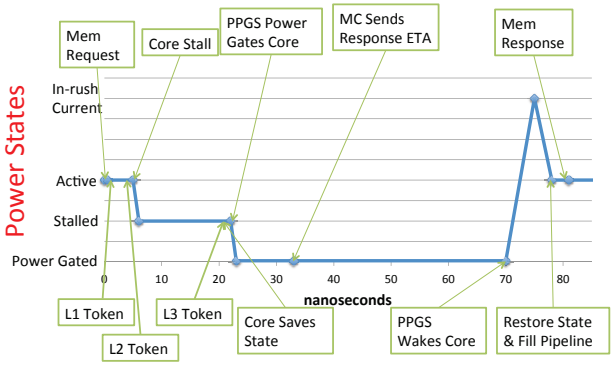

Figure 6: Timing diagram of TAP power gating a core.

core stalls due to a dependency. After the L3 Token is received, the PPGS decides to power gate the core and saves all core state. The core is then power gated and the memory controller (MC) sends an updated ETA for the memory response. At 70ns, the PPGS begins waking up the core. At 78ns, the core state is restored and the pipeline is restarted. The memory response comes back at 81ns and the core resumes execution as if nothing happened.

5. RESULTS

Table 3 summarizes all system values used in our experiments. The system has 4 cores, each with its own private L1 and L2 caches, and a large shared L3 cache. The L3 cache forwards requests to the memory controller through a shared memory bus. The L1 and L2 cache configurations are 32KB-8way and 256KB-8way. The L3 cache is a relatively large, 8MB-16way, which we expect to minimize pressure on the memory subsystem and hence minimize gains we see from our power gating technique. For the core, we model an out-of-order DEC-Alpha EV6 with an issue width of six and clock frequency of 3.3GHz.

We simulate the system with the GEM5 simulator [4]. GEM5 is a full system simulator that can boot an unmodified OS. It features cycle-level models of an out-of-order core, the cache hierarchy, and the interconnect. We integrate GEM5 with DRAMSim2 [2] to provide cycle-level modeling of the memory subsystem including the memory controller, DRAM modules, and shared channels used for communication. We modify GEM5 to support our power gating token methodology described in Section 4. We simulate our system with 21 of the Spec2006 benchmarks using the simpoint methodology [15] in which 100M-instruction representative regions of execution are determined for each benchmark. To simulate each region, we fast-forward to 100M instructions before the region, warm-up the memory and caches, and then perform the detailed simulation.

Once simulation is complete, we feed the system configuration and performance counters to McPAT [12] to model power consumption. McPAT is comprised of a power, area, and timing framework that provides off-line power and area estimates for full systems designed in technology nodes between 90nm and 16nm. McPAT generates values for dynamic power, leakage power, peak power, thermal design power, and area. We update McPAT's *technology.cc* file to accurately reflect the ITRS 2010 update report [1].

Comparison to DVFS. We compare TAP to dynamic voltage and frequency scaling (DVFS). We calibrate our DVFS settings to match those of [21] for the 32nm technology node, in which a 7.5% reduction in voltage follows each 20% reduction in frequency. To direct the DVFS policy, we apply the technique from [20], which uses a cycle-per-instruction based metric, μmean, to detect memory bounded phases of execution. During execution, we sample the application's μmean to determine the most aggressive DVFS setting that may be used to save energy while sustaining at most a 5% performance hit. In addition, we also consider an *oracle* DVFS technique that chooses the DVFS point that results in the lowest energy delay product (EDP). This technique takes an arbitrary performance hit as long as more energy is saved. For both policies, we model the availability of five DVFS modes which include 100%, 95%, 90%, 80%, and 60% frequency.

parameter	value	notes
Core Model	DEC-Alpha EV6	
Core clock	3.3GHz-1.9GHz	
Execution	6-way out-of-order	
Functional Units	6ALU,2IMULT 2FPALU	
ICache/Dcache	32KB-8way 1cyc	
L2 Cache	256KB-8way 4ns	Private per core
L3 Cache	8MB-16way 13ns	Shared
Memory	DDR3 2GB 50ns	
Core-to-L1 token latency	0.5ns	controller delays
Core-to-L2 token latency	4.5ns	controller delays
Core-to-L3 token latency	17.5ns	controller delays
Core-to-WUC latency	5ns	controller delays
PPGS wake-up modes	4.5ns-16.9ns	SPICE
EV6 pipeline refill latency	2.12ns	7 pipeline stages
EV6 core wake-up energy (EWE)	15,358pJ	Charge cells
EV6 leakage power (ELP)	0.916 Watts	McPAT [12]
EV6 PG leakage reduction (EPLR)	97.65%	[5]
EV6 PG break even point	17.17ns	$EWE/(EPLR*ELP)$
EV6 DFLT core wake-up latency	10.2ns	SPICE
FUPG wake-up energy	9641pJ	McPAT, ITRS [1]
FUPG wake-up latency	6.4ns	SPICE

Table 3: System Configuration Values

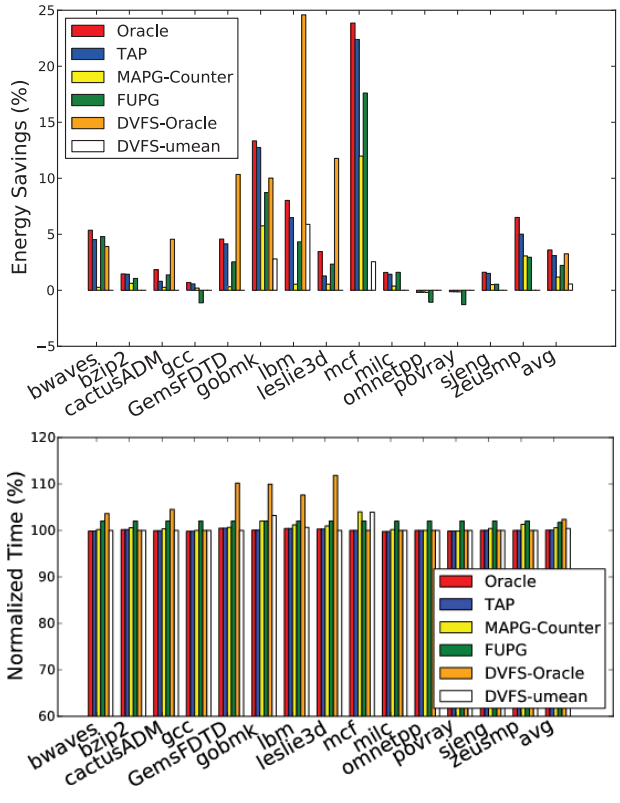

Figure 7: Energy savings and performance overhead of power gating Oracle, TAP, MAPG-Counter, FUPG, DVFS-Oracle and DVFS-μmean. Benchmarks with less than 1% absolute change were filtered out for readability

Comparison to MAPG. We also compare TAP to MAPG-Counter [9]. This scheme works by observing core stall time and predicting its duration. If the core stalls for longer than the average hit latency of the last-level cache, then MAPG-Counter predicts that the stall will last for the latency of a row buffer miss plus δ, where δ is an exponential weighted moving average which adapts to variations in the memory latency. MAPG-Counter uses this predictor to direct its power gating strategy.

For our reported results, we assume a four-core system that is 50% utilized (two cores idle) in a 32nm technology. When determining energy and performance values, we consider core wake-up energy, core wake-up delay, core pipeline refill latency, retention overhead of live-slave retention cells, SRAM leakage during source biasing mode of operation, core-to-WUC communication overhead, staggered wake-ups, and PDN voltage noise safety.

Comparison to Oracle Prediction. Figure 7 compares the energy savings of TAP with the energy savings of an oracle memory predictor (Oracle), MAPG-Counter [9], Functional Unit Power

Gating (FUPG) [8, 13, 14], DVFS-Oracle, and DVFS-μmean. To understand the limit of energy savings from power gating cores during memory stalls, the oracle memory predictor assumes *a priori* knowledge of all memory accesses and latencies and determines the optimal power gating behavior. The EV6 oracle policy achieves a maximum of 23.9% energy savings, and 3.6% savings on average. A few benchmarks show negative energy savings as high as -0.2%. These negative energy savings are caused by the lack of power gating opportunities and the retention cells' power overhead on cpu-bound benchmarks.

In comparison with the Oracle, TAP must determine memory latencies in a running system to ensure that sufficient time is available to power gate a core. TAP EV6 is able to achieve 22.4% (23.9% is Oracle) maximum energy savings, and 3.10% on average. TAP does not achieve the same energy savings as the Oracle because TAP is not able to power gate memory accesses until they miss in the L3 cache, and because lower-bound latencies are used. The result is that TAP avoids any performance hit but misses out on power gating at the beginning of the core stall. TAP also sees a few benchmarks with -0.2% energy savings due to cpu-bound behavior.

MAPG-Counter [9] power gates cores after the core stalls for longer time than the L3 hit latency, and power gates for the predicted stall duration according to its exponential learning algorithm. MAPG-Counter EV6 is able to achieve up to 12.0% energy savings (1.2% savings on average). TAP is able to achieve 2.58$\times$ the average energy savings of MAPG-Counter for out-of-order cores. (If method *A* saves 1% energy on average, and method *B* saves 3% energy on average, we say that "method *A* achieves 3.00$\times$ the average energy savings of method B".) The reason for this finding is that out-of-order cores stall more randomly than in-order cores, which makes the adaptive counter mechanism unstable and more prone to misprediction.

FUPG EV6 has a maximum and average energy savings of 17.6% and 2.2% with a maximum performance hit of 2% (1.5% average), at which point control logic prevents future power gating actions. The FUPG mechanism does better on average for the out-of-order core than MAPG-Counter, but TAP achieves 1.4$\times$ the average energy savings of the FUPG mechanism. FUPG does achieve more energy savings than TAP on a few cpu-bound integer codes in which not all the functional units are being used, but the core does not go idle. Greater energy savings could result from the cooperation of FUPG and TAP.

We also examine DVFS-Oracle using the scaling properties described in Section 5. The maximum and average EV6 core energy savings are 24.6% (lbm) and 3.3%, respectively. DVFS-Oracle on an EV6 core sees less maximum and average energy savings compared to the power gating oracle, but greater savings when compared to TAP. However, these energy savings suffer from two shortcomings. First, DVFS-Oracle has a maximum and average performance hit of 11.8% (GemsFDTD) and 2.4%. Second, these energy savings rely on oracle knowledge.

To understand DVFS under a realistic state-of-the-art policy, we consider DVFS-μmean [20], which predicts the performance hit of DVFS at each interval based on performance counters. DVFS-μmean achieves a maximum and average energy savings of 5.9% and 0.6% respectively. Thus, DVFS-μmean achieves less energy savings than TAP while experiencing a 1.0% performance hit on average (3.9% maximum). This result highlights the challenge of effectively applying DVFS at 32nm to match different application behaviors, and why TAP's determinism can result in higher energy savings.

6. CONCLUSION

With each successive generation of microprocessors, leakage power becomes an increasingly dominant issue. TAP effectively reduces wasted leakage power for out-of-order cores waiting for the memory subsystem. TAP achieves 22.39% maximum energy savings with no performance hit for out-of-order cores. Our method also achieves 2.58$\times$ the average energy savings of MAPG-Counter and 5.17$\times$ the average energy savings of a practical DVFS policy. Further, we are able to produce an accurate model of safe wake-up modes for an arbitrary arrangement of core states (waking-up, idle, active) in any arrangement of locations; this enables more aggressive wake-up modes for out-of-order cores. Our model reveals the importance of staggered wake-up as a technique to reduce voltage noise fluctuations across the power distribution network, as several cores try to wake up simultaneously. A staggered wake-up of 0.9ns between adjacent cores reduces core wake-up latency by up to 40.3%.

7. ACKNOWLEDGMENTS

This research was supported by Oracle, NSF, the MARCO FCRP (MuSyC and GSRC centers), and Qualcomm.

8. REFERENCES

[1] *International Technology Roadmap for Semiconductors*, 2010, http://www.itrs.net/Links/2010ITRS/2010Update .

[2] *DRAMSim2*, 2011, http://www.ece.umd.edu/dramsim/ .

[3] K. Agarwal, H. Deogun, D. Sylvester and K. Nowka, "Power Gating with Multiple Sleep Modes", *Proc. ISQED*, 2006, pp. 633–637.

[4] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi and S. K. Reinhardt, "The M5 Simulator: Modeling Networked Systems", *IEEE Micro* 26(4) (2006) pp. 52–60.

[5] D. Flynn, R. Aitken, A. Gibbons and K. Shi, *Low Power Methodology Manual*, Springer, 2007.

[6] K. He, R. Luo and Y. Wang, "A Power Gating Scheme for Ground Bounce Reduction during Mode Transition", *Proc. ICCD*, 2007, pp. 388–394.

[7] M. Horiguchi, T. Sakata and K. Itoh, "Switched-Source-Impedance CMOS Circuit for Low Standby Subthreshold Current Giga-Scale LSI's", *IEEE JSSC* 28(11) (1993) pp. 1131–1135.

[8] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units", *Proc. ISLPED*, 2004, pp. 32–37.

[9] K. Jeong, A. B. Kahng, S. Kang, T. S. Rosing and R. Strong, "Memory Miss Power Gating", *Proc. DATE*, 2012, pp. 1054–1059.

[10] R. Kumar and G. Hinton, "A Family of 45nm IA Processors", *IEEE ISSCC*, 2009, pp. 58–59.

[11] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan and C. Kozyrakis, "Power Management of Datacenter Workloads using Per-Core Power Gating", *IEEE Computer Architecture Letters* 8(2) (2009) pp. 48–51.

[12] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen and N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures", *Proc. MICRO*, 2009, pp. 469–480.

[13] A. Lungu, P. Bose, A. Buyuktosunoglu and D. J. Sorin, "Dynamic Power Gating with Quality Guarantees", *Proc. ISLPED*, 2009, pp. 377–382.

[14] N. Madan, A. Buyuktosunoglu, P. Bose and M. Annavaram, "A Guarded Power Gating for Multi-Core Processors", *Proc. HPCA*, 2011, pp. 291–300.

[15] E. Perelman, G. Hamerly, M. Van Biesbrouck, T. Sherwood and B. Calder, "Using SimPoint for Accurate and Efficient Simulation", *Proc. Intl. Conference on Measurement and Modeling of Computer Systems*, 2003, pp. 318–319.

[16] H. Qin, Y. Cao, D. Markovic, A. Vladimirescue and J. Rabaey, "SRAM Leakage Suppression by Minimizing Standby Supply Voltage", *Proc. ISQED*, 2004, pp. 55–60.

[17] Y. Shin, J. Seomun, K.-M. Choi and T. Sakurai, "Power Gating: Circuits, Design Methodologies, and Best Practice for Standard-Cell VLSI Designs", *ACM Trans. on Design Automation of Electronic Systems* 15(4) (2010) pp. 1–37.

[18] H. Singh, K. Agarwal, D. Sylvester and K. Nowka, "Enhanced Leakage Reduction Techniques Using Intermediate Strength Power Gating", *IEEE Trans. on VLSI Systems* 15(11) (2007) pp. 1215–1224.

[19] H. Zheng and Z. Zhu, "Power and Performance Trade-Offs in Contemporary DRAM System Designs for Multicore Processors", *IEEE Trans. on Computers* 59(8) (2010) pp. 1033–1046.

[20] G. Dhiman and T. Rosing, "Dynamic Voltage Frequency Scaling For Multi-Tasking Systems Using Online Learning", *Proc. ISLPED*, 2007, pp. 207–212.

[21] M. Cho, N. Sathe, M. Gupta, S. Kumar, S. Yalamanchilli and S. Mukhopadhyay, "Proactive Power Migration to Reduce Maximum Value and Spatiotemporal Non-uniformity of On-Chip Temperature Distribution in Homogeneous Many-Core Processors", *Proc. Semiconductor Thermal Measurement and Management Symp.*, 2010, pp. 180–186.

[22] O. Wechsler, "Setting New Standards for Energy-Efficient Performance", *Technology@Intel Magazine*, 2006.