MAPG: Memory Access Power Gating

Kwangok Jeong[1], Andrew B. Kahng[1,2], Seokhyeong Kang[1], Tajana S. Rosing[1,2] and Richard Strong[2]

[1] Dept. of Electrical and Computer Engineering, UC San Diego, La Jolla, CA USA
[2] Dept. of Computer Science and Engineering, UC San Diego, La Jolla, CA USA

***Abstract*—In mobile systems, the problems of short battery life and increased temperature are exacerbated by wasted leakage power. Leakage power waste can be reduced by power-gating a core while it is stalled waiting for a resource. In this work, we propose and model *memory access power gating* (MAPG), a low-overhead technique to enable power gating of an active core when it stalls during a long memory access. We describe a programmable two-stage power gating switch design that can vary a core's wake-up delay while maintaining voltage noise limits and leakage power savings. We also model the processor power distribution network and the effect of memory access power gating on neighboring cores. Last, we apply our power gating technique to actual benchmarks, and examine energy savings and overheads from power gating stalled cores during long memory accesses. Our analyses show the potential for over 38% energy savings given "perfect" power gating on memory accesses; we achieve energy savings exceeding 20% for a practical, counter-based implementation.**

I. INTRODUCTION

In mobile devices, operation time and peak processor performance are limited by battery capacity and chip thermal limits. These limits demand that all available power is used as efficiently as possible. However, a significant portion of power usage is leakage power. At the $32nm$ and $22nm$ technology nodes, leakage power ranges from 16.9% to 52.7% of total core power depending on circuit type, latency constraints, and temperature [1]. This leakage power translates into significant wasted energy if a core stalls waiting for a resource.

A core may stall quite often if it is intensely accessing the memory subsystem: 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 stall during which no forward thread progress occurs. Indeed, five of the Spec2006 [2] benchmarks (*GemsFDTD*, *gobmk*, *lbm*, *mcf*, and *milc*) spend more than 50% of their execution time waiting for the memory subsystem. Increased memory pressure in multicore processors suggests that waits will become longer as more threads contend for the memory resource. Power-gating the core during a memory access can potentially mitigate costly leakage power dissipation during core stalls.

Power gating [3] drastically reduces leakage power by introducing a switch between the voltage supply (and/or ground) and a given block of functional circuitry; the block's leakage is stopped when the switch cuts off the current path from supply to ground. The area and capacitance of a power-gated circuit block, as well as the supply noise tolerance of neighboring blocks that share supply rails with the gated block, determine several key parameters of power gating: block wake-up latency, block wake-up energy, peak ("inrush") current on wake-up, and supply voltage noise caused by wake-up. Modern architectures only power gate cores with long idle periods (on the order of $100ms$ [4]), because the core wake-up latency limits the applicability of power gating for short time intervals. [6], [7] have proposed multi-mode power gating circuits that offer variable wake-up latency, but these works are directed at the functional block level and focus on the tradeoff between leakage energy savings and wake-up latency. Our motivating observation is that if we can achieve power gating wake-up latency less than the expected memory access latency, much of the wasted leakage power can be eliminated.

In this paper, we propose a new system, *Memory Access Power Gating* (MAPG), which does not wait for threads to finish execution, but instead applies power gating during core stalls which are caused by the variable latency of requests to the memory subsystem. Our work makes the following contributions:

- We design and apply a two-stage wake-up power gating switch [8] to control core wake-up peak current, voltage noise, and latency without sacrificing leakage energy savings.
- We extend the two-stage power gating switch to have 10 wake-up modes by controlling the number of power-gated switches turned on in the first stage.
- We demonstrate the potential to reduce core energy consumption by up to 38.07% using our power gating switch and *oracle* knowledge of memory stalls.
- We design a practical, *counter-directed* power gating controller that can predict power gating windows and reduce core energy consumption by up to 22.47%.
- We develop a methodology to model voltage noise constraints across a power delivery network that apply to cores with wake-up latency on the order of 10 nanoseconds.
- We determine *safe* core wake-up modes for CMPs of 2, 4, 6, and 8 cores that consider core wake-up location, system utilization, and neighbor core supply voltage noise limits.

II. RELATED WORK

Power gating technology is already visible in leading commercial products. The recent Nehalem architecture employs power gating at the core level to reduce leakage power on idle cores [9]. Leverich et al. [4] have proposed that the operating system power-gate cores after threads block from a long IO operation, taking advantage of long idle periods on the order of $100ms$.

In the realm of architectural-level power gating, Hu et al. [10] 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 (ideal, time-based, and branch-misprediction-guided) predictors. 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, but functional units only make up a portion of total core leakage power.

Lungu et al. [11] show that Hu et al.'s predictor can increase energy consumption. [11] introduces a monitor that controls power gating to bound performance and energy penalty from misbehaved applications. Madan et al. [12] extend the ideas of [11] and propose a "guard mechanism" to reduce harmful use of power gating.

In the realm of circuit innovation, the recent survey of Shin et al. [3] summarizes the history of power gating techniques. We note that

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). Kwangok Jeong (kwang@samsung.com) is currently with Samsung Electronics Corporation, Yongin City, Korea.

978-3-9810801-8-6/DATE12/©2012 EDAA

configurable power gating has been introduced in the past to mitigate process variation, reduce ground bounce noise, and minimize wake-up latency. Agarwal et al. [6] and Singh et al. [7] 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 previous works do not consider the overhead of sleep control signal distribution.

Kim et al. [13] propose a tri-mode power gating structure in which a PMOS switch is combined in parallel with traditional NMOS power gating switches. The additional PMOS transistor supports intermediate power-saving state-retaining modes at low supply voltage, and reduces ground bounce noise during transitions between normal and power-gated modes. Chowdhury et al. [14] propose a similar tri-mode (i.e., RUN, HOLD, CUT-OFF) power gating technique using PMOS switches in parallel with NMOS footer switches, combined with additional NMOS switches in parallel with PMOS header switches. Finally, Zhang et al. [15] propose a multi-mode power gating technique using three NMOS switches with different sizes and threshold voltages. Using various combinations of the three switches, [15] provide multiple power gating modes with different leakage savings, and achieve improved tolerance to process variations.

To the best of our knowledge, previous techniques trade off faster wake-up modes for less leakage savings and lack explicit control of wake-up time and current profiles. Further, a common gap in previous works is that they do not incorporate a realistic power delivery network (PDN) model. By contrast, we explicitly control both wake-up latency and current profile, and we achieve shorter wake-up latency without sacrificing leakage savings; we also determine safe wake-up latencies based on realistic PDN modeling consistent with leading-edge mobile products at the 28nm foundry half-node.

III. MAPG System Design

In this section, we first introduce power gating and discuss the design of our programmable power gating switch (PPGS). We then discuss modeling and calculation of core capacitance and safe wake-up modes, and control of the PPGS.

A. Programmable Power Gating Switch (PPGS) Design

As noted above, power gating cuts off leakage current paths between supply (Vdd_core) and ground (Vss) by using switch transistors (often, high-Vth or long-channel devices). A typical power gating methodology with *header* switches is illustrated in Figure 1. When the pg_enable signal goes low, the header switches turn off and leakage current is reduced. While in the sleep (i.e., power-gated) state, all logic gates connected to the virtual supply (Vdd_int) lose their logical states. Setting the pg_enable signal to high resumes circuit operation after a delay that corresponds to charging circuit capacitive loads, resetting memory elements, and restoring state from retention flip-flops connected to Vdd_core.

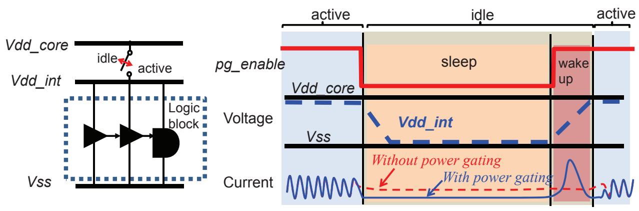
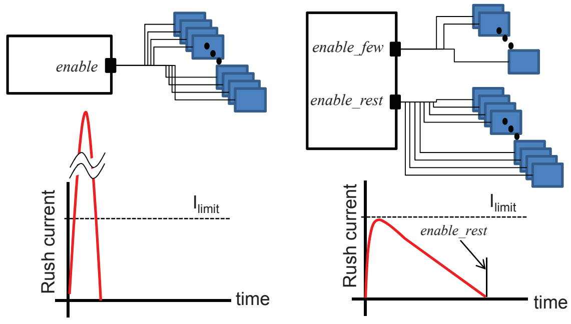


Fig. 1: Operation of the power gating technique.

The delay to charge circuit capacitive elements is a function of total design charge (Q) and available charging current (I_{limit}). If all header switches turn on simultaneously, a large "inrush" current charges internal nodes in minimal time. To satisfy inrush current upper limits (too-large IR drop can affect functionality of neighboring active blocks), header switches are partially turned on in sequence, which increases charging time to at least $T_{charge} = Q/I_{limit}$. Minimal charging time is achieved with a rectangular current profile, but such a profile requires very fine-grained control of header switches. To avoid this design complexity, we use a two-stage wake-up control [8] where the first stage (*enable_few* signal) turns on header switches to allow I_{limit} charge current. The remaining header switches are turned on in the second stage (*enable_rest* signal) once the circuit nodes are nearly charged, resulting in a triangular charging current profile (see Figure 2(b)). This acts to increase the wake-up latency to at least twice the minimum square wake-up profile, but significantly simplifies signal connections.



(a) Simultaneous wakeup (b) Two-stage wakeup

Fig. 2: Wake-up current profiles with different wake-up controls.

Figure 3 shows the interface between the core module and power gating controller. Inside the core, there are two power domains, collapsible and non-collapsible. During power gating, the non-collapsible domain provides power to retention registers and clamp circuits to allow data retention. (SRAM has its own voltage domain, and, e.g., source biasing by up to half of nominal supply voltage is used to reduce SRAM leakage during standby [16].)

To retain internal data during power gating, additional cycles are required for the power gating and wake-up sequence, as described in Figure 4. When power gating is triggered from a controller, the core clock is disabled before data retention. Then, the retention and the clamp signals are asserted to retain data. After data retention completes, switch enable signals are de-asserted and power switches are turned off. The wake-up sequence is the reverse order of the power-down sequence. To exit power gating mode, the *enable_few* signal is asserted to start the first stage of internal node charging. After the Vdd_int nodes are charged completely in time T_{charge}, the *enable_rest* signal is asserted. Asynchronous reset is asserted to initialize the internal states of the core. Then, the retention signal is de-asserted to restore data from live-slave retention flip-flops.[1] Additional cycles ($T_{restore}$) are required to restore data in other normal flip-flops (e.g., pipeline filling).

In order to maximize opportunities for power gating subject to wake-up inrush current and supply noise constraints, we seek to enable multiple wake-up modes, with a range of wake-up latencies, per core. Figure 5 shows our *programmable power gating switch* (PPGS) for a core, along with the wake-up current profile for different wake-up modes. We configure the number of first-stage wake-up switches to control the inrush current as shown in Figure 5(b). With the dynamic configuration of the PPGS, we can minimize the wake-up time according to the core configurations — e.g., the number or location of active cores relative to the waking-up cores. When we power-gate a core, all mode selection signals $m[0-9]$ are set to one,

---

[1] The live-slave type retention register retains data in its slave latch, and has smaller area and leakage overhead compared to the balloon type retention register [17].

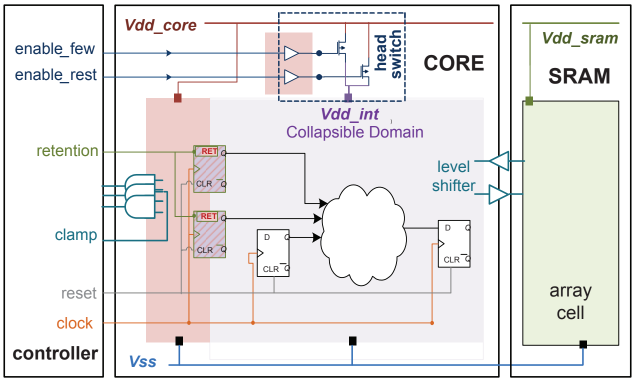


Fig. 3: Interface between core and power gating controller.

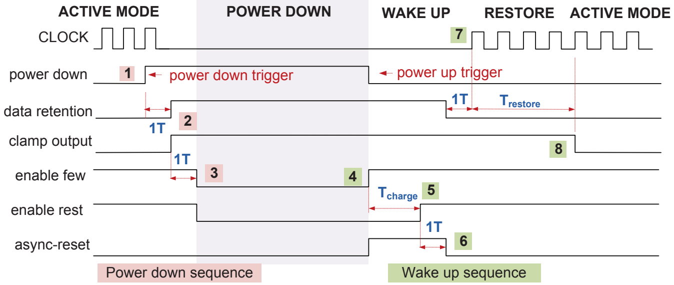
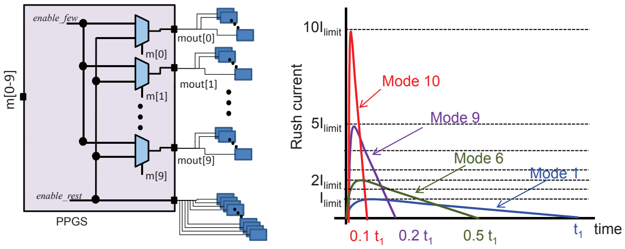


Fig. 4: Power gating and wake-up sequences.

which turns off all switches at the same time.[2]

In our PPGS design, the wake-up time and inrush current are determined by the mode selection. For example, Mode 1, which has the slowest wake-up time and lowest inrush current, is set by $m[0]$ = 0 and $m[1-9]$ = 1. Thus, $m[0]$ is enabled by signal *enable_few* and $m[1-9]$ is enabled by signal *enable_rest*. Mode 2 is set by $m[0-1]$ = 0 and $m[2-9]$ = 1; inrush current increases with the number of first-stage switches, while wake-up time decreases, as shown in Figure 5b. The other modes can be set similarly.



(a) PPGS control layout

(b) PPGS inrush current profile

Fig. 5: PPGS design and wake-up current profiles (modes).

B. *Calculating Minimal Safe Wake-up Modes for a Core*

Table I shows estimated design parameters, power gating results and PDN model parameters for 32nm and 22nm cores with high performance (HP) and low-operating power (LOP) devices. To study wake-up latency and inrush current, we estimate the total charge for core logic and interconnect capacitance as Q_{core} = $(C_{logic} + C_{int})\,Vdd\_core$, where Q_{core}, C_{logic}, and C_{int} represent total charge, device capacitance, and interconnect capacitance for a single core without caches. We estimate that the $EV4$ core has 6.25M transistors from McPAT [1] data for both the 32nm and 22nm designs. Based on this transistor count and parameters from the 2009-2010 *International Technology Roadmap for Semiconductors* (ITRS) [18], we estimate C_{logic} and C_{int}. The inrush current limit (I_{limit}) and on-current (I_{active}) are estimated from McPAT data for peak power and average power, respectively.

From the calculated charge (Q_{core}), the minimum wake-up latency with a rectangular-form current profile is $T_{min-charge}$ = Q_{core}/I_{limit} and the minimal two-stage wake-up latency (Figure 2(b)) is $2 \times T_{min-charge}$.

We estimate leakage power consumption during power gating of the core logic and SRAM. For the core logic, leakage from retention registers and header switches must be taken into consideration. We assume that retention flip-flops have 20% more leakage power than normal flip-flops during power gating [17]. We use retention registers for the first pipeline stage.[3] For SRAM, we assume that the (separate) SRAM supply voltage is scaled using source biasing, and we estimate leakage based on [16].

TABLE I: Estimated data of 32nm HP, LOP and 22nm HP, LOP cores.

Estimated Data	32nm HP	32nm LOP	22nm HP	22nm LOP
Design Data				
Vdd_core (V)	1.00	0.77	1.00	0.77
core area (mm^2)	4.593	4.608	2.701	3.657
logic area (mm^2)	2.891	2.863	1.635	1.636
C_{core} (F)	7.53E-9	7.48E-9	4.58E-9	4.58E-9
total charge (C)	7.53E-9	5.76E-9	4.26E-9	3.30E-9
core leakage (W)	0.355	0.042	0.147	0.019
I_{active} (A)	0.725	0.374	0.371	0.233
I_{limit} (A)	1.298	0.674	0.701	0.632
Power Gating and Wake-up				
$T_{min-charge}$ (ns)	5.08	7.36	6.40	6.55
wake-up energy (pJ)	3.30E+3	1.91E+3	2.24E+3	1.60E+3
# head switches	9,664	6,222	5,516	5,127
leakage in PG state (W)	8.03E-3	7.14E-4	3.37E-3	3.59E-4
leakage reduction in PG	97.74%	98.29%	97.71%	98.12%
PDN Model				
# bump	45	45	95	95
R_{shared} (Ω)	0.01	0.01	0.01	0.01
$L_{pkg-core}$ (nH)	7.69E-4	7.76E-4	6.44E-4	6.44E-4
$R_{pkg-core}$ (Ω)	1.54E-5	1.55E-5	1.29E-5	1.29E-5
C_{decap} (F)	1.51E-9	1.50E-9	9.16E-10	9.16E-10
R_{PDN} (Ω)	0.07	0.10	0.12	0.15

We construct a detailed PDN model that includes 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 shown in Figure 6. 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 package 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 (see Figure 6). On-chip decoupling capacitance C_{decap} is assumed to be 20% of C_{core} as in Huang et al. [19].

PDN parameter values in Table I are from personal communication with industry experts [20] and reflect production designs at the 28nm foundry half-node. Bump density is assumed to be 45 bumps per mm^2, and the number of bumps is then calculated from logic area

---

[2] Due to the large resistance of off-state switches, inrush current from simultaneous turnoff is negligibly small compared to wake-up inrush current.

[3] We must wake up the core 4 cycles earlier to fill the pipeline.

TABLE II: Wake-up latency (ns) for each PPGS wake-up mode ($32nm$ HP).

mode	*1*	*2*	*3*	*4*	*5*
latency(ns)	14.16	13.14	12.12	11.11	10.09
mode	*6*	*7*	*8*	*9*	*10*
latency(ns)	9.08	8.06	7.05	6.03	5.02

TABLE III: Minimum safe wake-up mode with respect to the number of idle cores for a $32nm$ HP processor.

# idle cores	0	1	2	3	4	5	6	7
2-core	4	10	-	-	-	-	-	-
4-core	2	2	7	10	-	-	-	-
6-core	1	2	3	4	8	10	-	-
8-core	2	2	2	2	3	5	8	10

(I/O signals are peripherally located in the SoC die plan). The package inductance and resistance to a bump are respectively assumed to be $0.05nH$ and $1m\Omega$ based on the empirical data. The lumped package inductance $L_{pkg-core}$ and resistance $R_{pkg-core}$ for a single core are respectively calculated as L_{pkg}/N_{bump} and R_{pkg}/N_{bump}, where N_{bump} is the number of bumps.

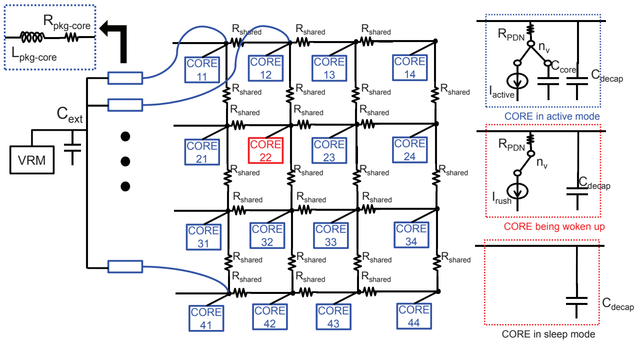


Fig. 6: 16-core system power delivery network with power gating.

We measure the Vdd_core and Vdd_int voltages of all cores using HSPICE. We vary the number of cores being woken up, and search over all configurations of woken-up and active cores. For each configuration, we find the minimum wake-up latency that satisfies two IR drop constraints: (a) Vdd_int of active cores should drop by no more than 5% (threshold = $V_{low,virtual}$) and (b) Vdd_core of standby cores should drop by no more than 40% to retain data in retention circuits (threshold = $V_{low,core}$) [20].

Table II gives the wake-up latency for each of the 10 wake-up modes for a $32nm$ HP core. We use HSPICE simulation to check the IR drop constraints for each wake-up mode across all spatial configurations; Table III shows the best (minimum latency) wake-up mode that is safe for each number of woken-up cores. The tables confirm that as the number of active cores increases, the minimum safe wake-up latency increases. We exploit this information in PPGS mode control, and use the minimum safe mode to reduce the wake-up time.[4]

C. MAPG Controller Design

Our system also includes a MAPG Controller, which enables each core to power-gate on memory-induced stalls, and to wake up using the minimal safe wake-up mode consistent with system utilization. The MAPG Controller should also avoid any overhead in terms of extra execution time or energy. We can calculate the interval over which a core must be power-gated, so as to break even on wake-up energy costs — e.g., $Energy_{wake-up}/(Power_{core\_leakage} * PGLeakageReduction)$ from Table I as $9.5ns$.[5] To avoid both energy and performance overhead for a core wake-up delay of $8.06ns$, a core must receive no memory response for at least $17.6ns$ after the core is power-gated. Any power-gated time over $17.6ns$ yields energy savings. The expected response time at the core of a L3 cache hit is equal to $22ns$, but the core would not know to power gate for it until after the expected L2 hit time of $5ns$; this gives a predictable power gating window of only $17ns$. Thus, we do not try to power-gate L2 misses and instead focus on memory accesses that go all the way to the memory controller.

To aid the following discussion, we define the following core states:

- **Active:** a core which is either executing code or waking up from a power-gated state.
- **Stall:** a core that is not making forward progress on an application because of a memory dependence.
- **Idle:** a core with a thread that has either exited or blocked and will not be running more code for at least 100ms.

In order for the PPGS to power-gate a core, it must determine the wake-up mode and the interval over which to power-gate. Wake-up mode is determined by the number of *idle* cores from Table III. For instance, for a 4-core system, we see that wake-up modes 2, 2, 7 and 10 are used when 0, 1, 2, and 3 cores are *idle*, respectively. For the PPGS to get the current wake-up mode, it must register with the Wake-up Controller (WUC) and set whether it is *active* or *idle*. If the PPGS registers with the WUC as *active*, the WUC returns the wake-up mode defined in its lookup table indexed by the number of *idle* cores to all active cores. If the PPGS registers with the WUC as *idle*, the WUC will send the new wake-up mode to the remaining active cores. A core is not allowed to power-gate during a memory stall until it has registered with the WUC; this prevents any violation of voltage noise constraints.

The interval over which to power-gate should be optimized to avoid any performance overhead and to maximize power gating duration. Performance overhead is avoided by (i) predicting a long stall interval, and (ii) waking the core at the expected end of the stall interval. The long stall interval is predicted by counting the number of stalled cycles after a core memory access. If the number of cycles reaches beyond the expected latency at which the core would have received an L3 hit response ($22ns$), the PPGS calculates the expected arrival time of the memory response and power-gates the core if sufficient time will elapse to save energy. The expected arrival time is estimated as the memory's row buffer miss latency ($45ns$) plus a value, δ. We compute δ as an exponential moving average of the difference between the actual and expected response arrival times, with the extra condition that if the expected arrival time is greater than actual, δ is immediately set to the difference.[6] This extra condition avoids repeated late core wake-ups and performance hits that would otherwise have to wait for the exponential moving average. In the common case, the exponential moving average will adapt to variable memory latency caused by contention for the memory resource, optimizing the duration of the power gating window.

---

[4] We have separately studied the sensitivity of our safe wake-up mode modeling. Minimum safe modes as in Table III will depend on the specific values of PDN parameters, with highest sensitivities being to core resistance (R_{PDN}), number of bumps, and package inductance (L_{pkg}).

[5] When cores are waking up, there will be signal ripples until all nodes settle at their final valid values. In our $Energy_{wake-up}$ calculation, we ignore these ripple effects since the effective operating voltage (between virtual supply and ground) is small during the wake-up.

[6] The exact calculation:
(1) $diff = MemDelay_{actual} - MemDelay_{expected}$;
(2) if $(diff < 0)$ $\{\delta = \delta + diff\}$ else $\{\delta = 0.8\delta + 0.2 \cdot diff\}$.

IV. RESULTS

A. Simulation Methodology

We simulate a 4-core system, with key microarchitectural settings as given in Table IV. Each core has a private L1 and L2 cache and shares a large L3 cache. The L3 cache forwards requests to the memory controller through a shared memory bus. The L3 cache is relatively large, which we expect to minimize extra pressure on the memory subsystem and hence minimize gains that we see from our power gating technique. We choose an in-order dual-issue core both to model future energy-efficient many-core processors and to pressure any energy-saving scheme, as this core is considered quite energy-efficient. This system is similar to the Intel ATOM and ARM8 processors.

TABLE IV: Architectural configuration used for cores and basic system.

Core	ISA, Model	Alpha64, EV4
	Execution	In-order
	Clock	$2GHz$
	ICache, DCache	$32kB$-2way
	Width	2
	Functional Units	2IALU,1IMULT,1FPALU
Memory Hierarchy	L2Cache	$256kB$-4way $4ns$
	L3Cache	$8MB$-16way $13ns$
	Memory Latency	$50ns$
	Memory Size	$2GB$

We simulate the system with the M5 2.0 simulator [21]. M5 is a full system simulator that can boot an unmodified OS. It features cycle-level models of an in-order core, the cache hierarchy, IO, and interconnect. We have modified M5 to support MAPG's counter-based and oracle power gating, described in the next subsection.

Once simulation is complete, we feed the system configuration and performance counters to McPAT [1] 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.

To evaluate single-threaded execution, we simulate 19 SPEC CPU2006 [2] benchmarks. We use Simpoint-3.2 [22] to determine the most representative region of execution for each benchmark, fast-forward simulation to 100E6 instructions before the point, warm-up the core for 100E6 instructions, and then run for 100E6 instructions. Multi-threaded workloads are created by combining the SPEC CPU2006 benchmarks into homogeneous sets. Homogeneous sets of benchmarks capture the dominant portion of core power gating behavior, since the core's power gating behavior is mostly defined by local thread behavior. Heterogeneous sets of benchmarks sometimes act to increase memory contention and increase power gating opportunities; we omit these results due to space limits.

B. Power Gating Energy Savings and Overheads

We now present the results for MAPG in a running system and give a comparison between MAPG using an *oracle* predictor (MAPG-Oracle) of core stall periods, MAPG using a counting based mechanism (MAPG-Counter) described in Section III-C, and functional unit power gating (FUPG) similar to that proposed in [10] and [11].[7] We assume a four-core system that is 50% utilized (two cores idle) using core wake-up mode 7 (8.06ns) and 32nm technology. When calculating the reported results, 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, and voltage noise safety.

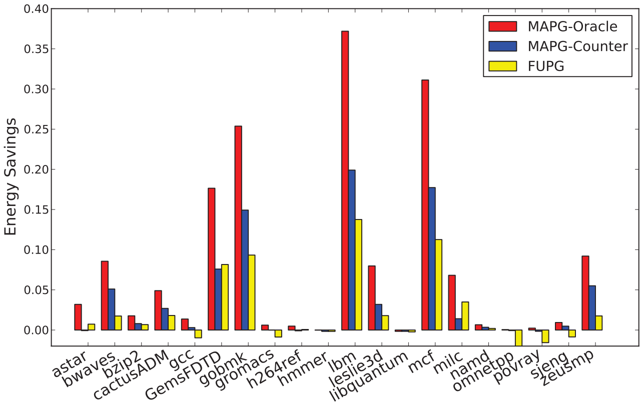


Fig. 7: Energy savings comparison of MAPG-Oracle and MAPG-Counter with 8.06ns wake-up delay, versus FUPG.

Ultimately, we are interested in how much energy we save by power-gating a core during a memory access. Figure 7 shows MAPG's energy savings for each of the SPEC CPU2006 benchmarks normalized to core execution without power gating or circuit support for power gating. For each benchmark, we present the energy savings for MAPG-Oracle, MAPG-Counter, and FUPG. First, MAPG-Oracle shows the limit of potential energy savings from MAPG, given oracle knowledge of core stall time. On average, the oracle system saves 8.80% energy across all benchmarks with a maximum energy savings of 38.07% for *lbm* and a minimum of -0.17% for *libquantum*. Greater energy savings occur when cores stall more often from accessing the memory; *lbm* is the most memory-bound benchmark and hence has the largest energy savings. The negative energy savings for *libquantum* are due to its cpu-bound behavior and the overhead from the power gating switches and retention flip-flops. MAPG-Counter is able to save 4.44% energy on average with maximum energy savings of 20.99% for *lbm* and a minimum of -0.17% for *libquantum* - for similar reasons as with MAPG-Oracle. FUPG saves 2.64% energy on average, with a maximum energy savings of 13.75% for *lbm* and a minimum of -1.54% for *povray*. FUPG's smaller energy savings are due to the fact that it power-gates only the functional units which make up 36.14% of core leakage power, and to the performance overhead from power gating short stall periods until the control logic prevents negative-energy power gating actions. Overall, MAPG-Counter is able to achieve better energy savings than FUPG in all cases except for benchmarks *astar* and *milc* for which FUPG power gates underutilized functional units.

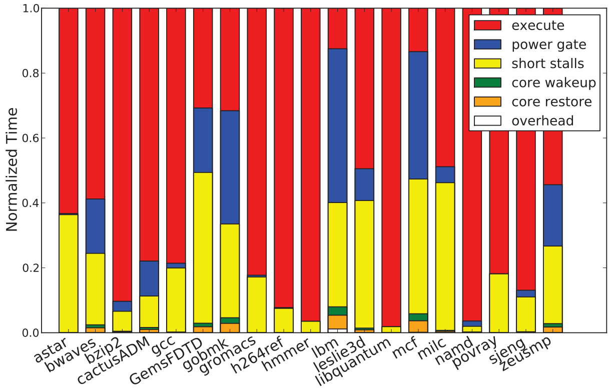


Fig. 8: Breakdown of simulation time for each benchmark utilizing MAPG-Counter.

Figure 8 examines simulation time of each benchmark and breaks it into time spent executing (execute), time spent power gating the core (power gate), short stalls that MAPG-Counter could not power gate without energy loss (short stalls), core wake-up time to charge core logic (core wake-up), core restore time to restore data from retentive flip-flops and fill the pipeline (core restore) and overhead added from waking up the core too late (overhead). Across all benchmarks, the average overhead added to execution time was 0.08%. Core restore and core wake-up latency averaged 0.60% and 0.99% of simulation time, respectively, with a maximum of 2.59% and 4.23% of the simulation time of *lbm* caused by the many power gating actions. On average, the core was power-gated 11.16% percent of the time with up to 47.39% of simulation time being power-gated for the benchmark *lbm*. In addition, short stall time made up an average of 21.70% of simulation time, with a maximum of 46.46% for the benchmark *GemsFDTD*; this indicates that there could be significant energy savings gains if idle periods could be predicted earlier and core wake-up latencies reduced.

---

[7] Functional units account for 36.14% of core energy and are assumed to take 6ns and 1780pJ to wake up.

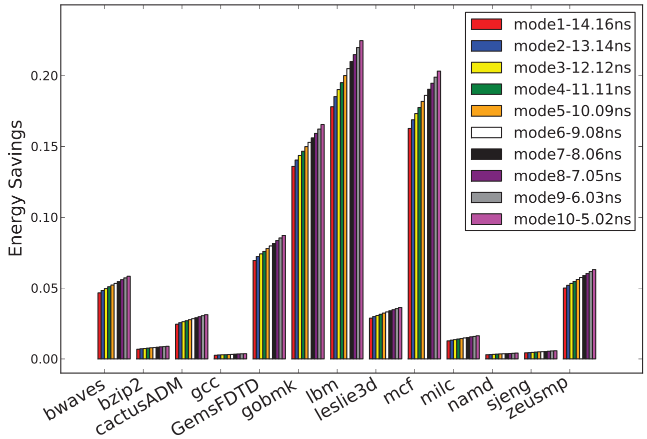


Fig. 9: Energy savings for MAPG as wake-up mode changes from mode 1 to mode 10. Benchmarks *astar*, *gromacs*, *h264ref*, *hmmer*, *libquantum*, and *povray* omitted due to small change (less than 0.2%) and space limits.

Finally, we examine the energy savings that will occur for different wake-up modes, to understand how much energy could be saved in a system that uses MAPG-Counter and dynamically adapts to wake-up latency. Figure 9 shows that energy savings increase linearly with reduced wake-up delay for the 13 benchmarks *bwaves*, *bzip2*, *cactusADM*, *gcc*, *GemsFDTD*, *gobmk*, *lbm*, *leslie3D*, *mcf*, *milc*, *namd*, *sjeng*, and *zeusmp*. Average energy savings increase from 3.78% at wake-up mode 1 to 4.75% at mode 10 across all benchmarks. The maximum energy savings increase occurs for *lbm* which goes from 17.81% energy savings at mode 1 to 22.47% energy savings at mode 10. The improved energy savings result from less wake-up time overhead and the ability to power-gate the core for longer periods. This means that an adaptive system could improve its energy savings by an additional 4.66% if mode 10 is a viable wake-up mode.

V. CONCLUSIONS AND FUTURE WORK

With each new generation of CPUs, leakage power becomes an increasingly dominant issue. In this paper, we have described our MAPG system, which effectively reduces wasted leakage power on cores that are waiting for the memory subsystem, without disturbing voltage noise safety constraints across the chip PDN. Specifically, MAPG-Oracle demonstrates the potential of the PPGS to reduce wasted core leakage power by 38.07% with oracle knowledge and a 8.06ns core wake-up delay. MAPG-Counter shows a realistic implementation of a PPGS controller that achieves energy savings as much as 20.99% for a 8.06ns wake-up delay. If allowed to dynamically adapt to system utilization levels, energy savings rise from 17.81% to 22.47% as the wake-up mode changes from 1 to 10.

Looking toward the future, Figure 8 indicates the importance of power-gating short stalls. If core wake-up delay can be significantly reduced and delay periods predicted sooner, the amount of time the core spends power gating can be greatly increased. In addition, we seek to provide a more accurate minimum wake-up latency via more detailed PDN modeling. The effect of MAPG on back-end-of-line interconnect reliability is currently an open issue. Finally, we recognize that it will be important to extend this work to out-of-order cores, as this would extend the scope of the technique and hopefully achieve greater energy proportionality in server processors; this is the subject of our ongoing efforts.

REFERENCES

[1] 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. IEEE MICRO*, 2009, pp. 469–480.

[2] *Standard Performance Evaluation Corporation*, 2006, http://www.spec.org/cpu2006 .

[3] 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.

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

[5] P. R. Panda, A. Shrivastava, B. Silpa and K. Gummidipudi, *Power-Efficient System Design 1st Edition*, Springer, pp. 25–73, 2010.

[6] K. Agarwal, H. Deogun, D. Sylvester and K. Nowka, "Power Gating with Multiple Sleep Modes", *Proc. International Symposium on Quality Electronic Design*, pp. 633–637, 2006.

[7] 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.

[8] K. He, R. Luo and Y. Wang, "A Power Gating Scheme for Ground Bounce Reduction during Mode Transition", *Proc. International Conference on Computer Design*, 2007, pp. 388–394.

[9] R. Kumar and G. Hinton, "A Family of 45nm IA Processors", *Proc. IEEE International Solid-State Circuits Conference*, 2009, pp. 58–59.

[10] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units", *Proc. International Symposium on Low Power Electronics and Design*, 2004, pp. 32–37.

[11] A. Lungu, P. Bose, A. Buyuktosunoglu and D. J. Sorin, "Dynamic Power Gating with Quality Guarantees", *Proc. International Symposium on Low Power Electronics and Design*, 2009, pp. 377–382.

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

[13] S. Kim, S. V. Kosonocky, D. R. Knebel, K. Stawiasz and M. C. Papaefthymiou, "A Multi-Mode Power Gating Structure for Low-Voltage Deep-Submicron CMOS ICs", *IEEE Trans. on Circuits and Systems II: Express Briefs*, 54(7) (2007) pp. 586–590.

[14] M. H. Chowdhury, J. Gjanci and P. Khaled, "Innovative Power Gating for Leakage Reduction", *Proc. IEEE International Symposium on Circuits and Systems*, 2008, pp. 1568–1571.

[15] Z. Zhang, X. Kavousianos, K. Chakrabarty and Y. Tsiatouhas, "A Robust and Reconfigurable Multi-mode Power Gating Architecture", *Proc. International Conference on VLSI Design*, 2011, pp. 280–285.

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

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

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

[19] G. Huang, D. Sekar, A. Naeemi, K. Shakeri and J. Meindl, "Physical Model for Power Supply Noise and Chip/Package Co-Design in Gigascale Systems with the Consideration of Hot Spots", *Proc. IEEE Custom Integrated Circuits Conference*, 2007, pp. 841–844.

[20] S. Dobre, Qualcomm CDMA Technologies, Inc., *personal communication*, March-Sept. 2011.

[21] 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.

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