Register Allocation and VDD-Gating Algorithms for Out-of-Order Architectures

Steven J. Battle and Mark Hempstead
Drexel University
Philadelphia, PA USA
Email: sjb328@drexel.edu, mark.hempstead@coe.drexel.edu

***Abstract*—Register Files (RF) in modern out-of-order microprocessors can account for up to 30% of total power consumed by the core. The complexity and size of the RF has increased due to the transition from ROB-based to MIPSR10K-style physical register renaming. Because physical registers are dynamically allocated, the RF is not fully occupied during every phase of the application. In this paper, we propose a comprehensive power management strategy of the RF through algorithms for register allocation and register-bank power-gating that are informed by both microarchitecture details and circuit costs. We investigate algorithms to control where to place registers in the RF, when to disable banks in the RF, and when to re-enable these banks. We include detailed circuit models to estimate the cost for banking and power-gating the RF. We are able to save up to 50% of the leakage energy vs. a baseline monolithic RF, and save 11% more leakage energy than fine-grained VDD-gating schemes.**

Index Terms*—*Computer architecture, Gate leakage, Registers, SRAM cells

I. Introduction

Out-of-order superscalar processors, historically found only in high-performance computing environments, are now used in a diverse range of energy-constrained applications from smartphones to data-centers. Despite active research in processor power management, a significant portion of active and static power is consumed by processors' register files. This occurs across computing domains; for example, the register file (RF) in the Motorola M-CORE embedded processor consumes 16% of total core power [1]. This consumption is exacerbated in modern high-performance out-of-order processors that have switched from ROB-based to MIPSR10K-based physical register-renaming. For example, the IBM POWER7 RFs consume 21% of core power, while the Intel Westmere RFs account for 30% of core power [2], [3]. An additional trend is the increasing contribution of static power to total microprocessor power consumption [4]. Again, the register file is a significant factor: the IBM POWER7 RF and Intel RF consume approximately 15% and 30% of core leakage respectively [2], [3]. Techniques such as VDD-gating [5], [6] and drowsy-modes [7], [8] have been used to address the energy-cost of register files on a fine-grained manner, while banked register files [9], [10] have been used to increase performance and reduce dynamic costs.

Register files in modern out-of-order processors must be large in order to support a large instruction window containing both architectural (committed) and speculative state; a bigger pool of rename registers eliminates false dependencies to support more instructions in flight. However, application phases do not always exhibit high ILP, often leaving a significant portion of the RF dormant.

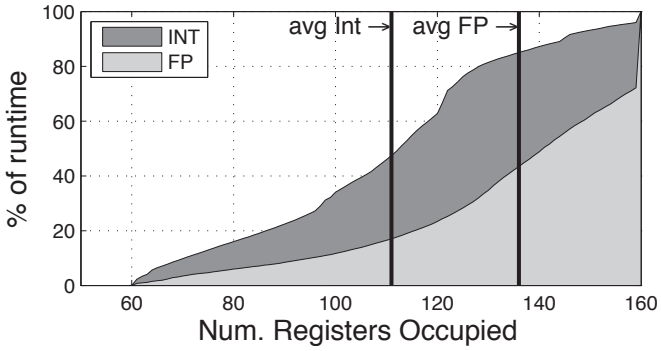

Fig. 1. **Average Reg File occupancy** CDF for SPEC2006 workloads.

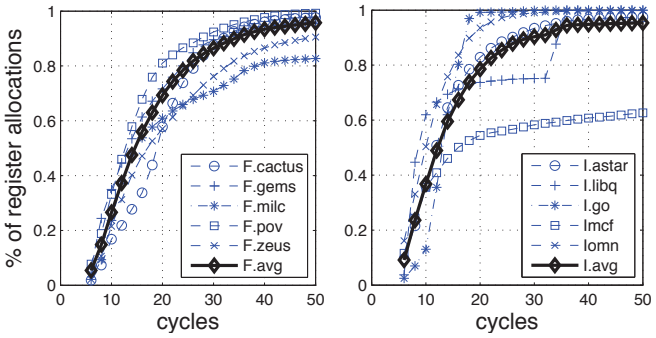

Fig. 2. **"Allocate-write distance"** CDF showing distance between register allocation at rename (cycle 0) and register use at writeback.

Figure 1 shows a histogram of register files occupancy across SPEC2006 benchmarks for a 160-entry register file modeled after Intel's Sandy Bridge architecture [11]. On average, only *68%* of the RF is in use for INT workloads and *78%* for FP workloads. In addition, even registers that are "occupied" do not always contain valid state. Figure 2 shows the the distance in cycles between register allocation at the register-rename stage and register-use during the writeback stage ("allocate-write" distance). A minimum of 6-cycles is needed between allocate and writeback yielding two slacks that can be exploited for energy reduction: slack in the *amount* of RF resources available, and slack in *timing* when a register needs to be available after allocation.

978-1-4799-2987-0/13/$31.00 ©2013 IEEE

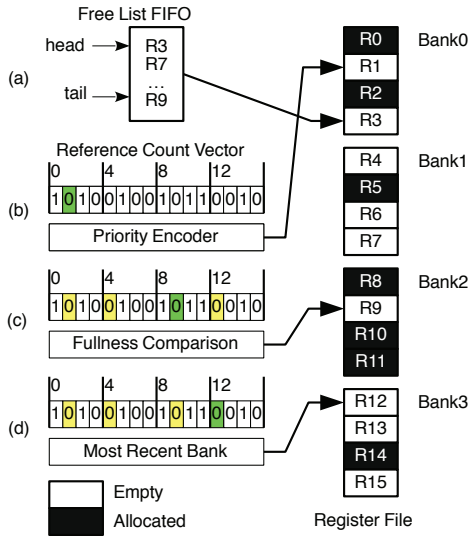

Fig. 3. **Allocation algorithms.** Each algorithm examines the set of available registers to select the next register for allocation. (a) Free-list: selects the reg at the head of the FIFO queue. (b) Prio: select first free reg from a bitvector representation of the RF ('0'=free, '1'=allocated). (c) Full: select first free reg from fullest RF bank. (d) MRA: select first free reg from most-recently-selected bank

We explore allocation and gating algorithms that are coupled with microarchitectural information to reduce RF energy usage. We present several algorithms using information such as instruction type, ROB activity, and register bank fullness to make allocation and VDD-gating decisions. We study their efficacy in reducing energy compared to a monolithic register file using conventional free-list allocation. In section IV, we review RF design and the circuit-costs of VDD-gating. In section V, we present our register allocation and RF-bank VDD-gating algorithms, with results and analysis in section VI.

II. Register Allocation

Physical registers are allocated to instructions during the rename stage of the out-of-order pipeline. An allocation algorithm examines the set of free registers, providing one to the next dispatching instruction. Modern out-of-order processors implementing MIPS-style register renaming typically manage registers using a circular queue free-list to identify unallocated registers [12]. Dispatching instructions are allocated a destination register from the head of the free-list (a dequeuing or "pop" operation). When an instruction commits its value to the architected state, the overwritten register is freed (enqueued or "pushed") to the tail of the free-list.

Figure 3 illustrates how register allocation affects register distribution: a 16-entry RF is shown with allocated registers shaded. Four allocation schemes, (a)-(d), select a different register according to their algorithm definition. (a) is conventional *free-list* allocation where the "next" register allocation is determined by the contents of the FIFO head-pointer. This leads to registers being distributed across the RF as the program executes. A *priority* encoded scheme is shown in (b), where register occupancy is represented by a bitvector [13]. The first empty bit in the vector is selected for the next

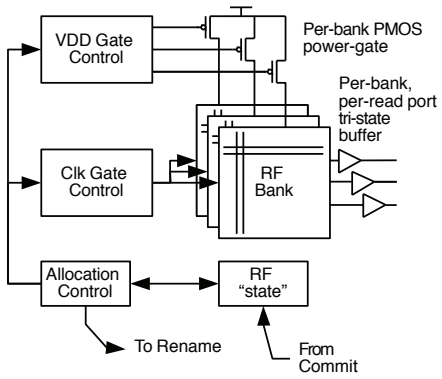

Fig. 4. **Banked Register File**: A partitioned RF allows for modular control of allocation, clock-gating, and VDD-gating when RF state is kept on a per-bank basis. Register reference counts monitor the RF and allow us to implement various allocation algorithms.

allocation, keeping newly allocated registers clustered to one end of the RF. In (c), bank occupancy is compared to select the first free register from the *fullest* bank, while in (d), the first free register from the most-recently selected bank is allocated.

III. Methodology

We evaluate our register file allocation and gating algorithms using both performance and circuit simulation. Our cycle-level performance simulator executes user level x86_64 code, breaking x86 instructions into RISC-like three-register micro-operations. The simulated core is modeled roughly after Nehalem. It is 4-wide issue with a 23-stage pipeline, 128-entry reorder buffer, 36-entry issue queue, and 96 rename registers. We model a single-threaded configuration with a register-file containing 160 registers, as described in Table I. Our circuit models are built using the NCSU FabScalar memory generator [14] in NSCU's 45nm FreePDK CMOS technology. We include measurements from HSPICE simulations to dynamically model power with our performance simulator.

We compiled all SPEC2006 benchmarks and simulate the benchmarks to completion on their training inputs, sampling 10 million of every 500 million instructions with 10 million instructions of cache and branch predictor warm-up. Note that while our figures omit benchmarks due to space, all benchmark data is included in average INT and FP columns.

IV. Register-File VDD-Gating Costs

When choosing a register file power management strategy, it is important to ensure that the circuit costs of toggling registers do not consume more energy than they save. This section describes our circuit models and estimation of VDD-gating overheads on the register file. VDD-gating is a circuit technique that can dramatically reduce leakage energy component by adding a PMOS gate transistor between the VDD power-rail and the logic circuit [5], [15]. VDD-gating is a destructive operation and only empty registers or registers whose contents are known to be expired may be gated. VDD-gating requires a PMOS gate-transistor, driver, and additional isolation circuitry to ensure un-gated logic is unperturbed. The cost to switch these circuits *must* be recovered by the leakage

Component	Value
Registers	160x64-bit
Area	70114 um^2
Ports	6 read, 3 write
E_{Read}	10.8 pJ
E_{Write}	15.1 pJ
Latency	2 cycles

TABLE I
BASELINE RF PARAMETERS

Component	I_{leak}	%
Precharge	174 uA	31.67%
SRAM	152 uA	27.67%
Buffers	142 uA	25.95%
WordLine	76 uA	13.90%
Decoders	2.57 uA	0.47%
SenseAmp	1.90 uA	0.35%

TABLE II
RF LEAKAGE COMPONENTS

Bank Size	W_{PMOS1}	T_{BE1}	W_{PMOS2}	T_{BE2}	EPC_2/EPC_1
1 (bit-cell-only)	0.180 μm	15	–	–	–
4	12.5 μm	21	6.5 μm	23	0.66
8	18.0 μm	23	9.0 μm	24	0.69
16	20.0 μm	21	10.0 μm	21	0.70

TABLE III
OVERHEAD AND BREAK-EVEN FOR PMOS GATES

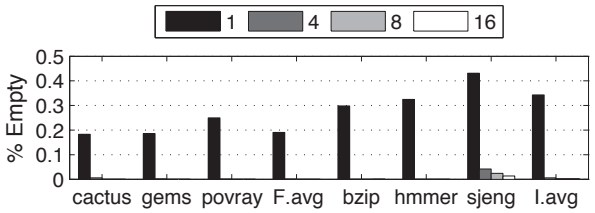

Fig. 5. **% RF Empty** vs. banksize for free-list allocation. When bank size > 1, the contiguous bank must be un-allocated to be considered 'Empty'.

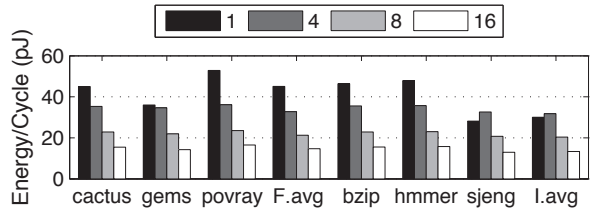

Fig. 6. **Total RF energy/cycle** vs. bank size for freelist allocation assuming empty banks can be VDD-gated

energy reduction in order to break-even and be advantageous compared to clock-gating, which has no intrinsic circuit cost to the RF itself.

There are two approaches to VDD-gating of register files: fine- and coarse-grained. Several previous studies have focused on fine-grained gating of individual registers, but without detailed analysis of the energy, performance, or area costs associated with such fine-grained partitioning. Goto and Sato proposed a dynamic gating algorithm using free-list allocation in out-of-order processors, toggling individual registers when they are enqueued and dequeued from the free-list [16]. Khasawneh and Ghose proposed an adaptive technique to disable registers in two places: when the register is allocated but has not been written, and when the register has been both written and consumed but not de-allocated [17]. Battle et al. introduced the concept of using reference counts for coarse-grained register file gating, but only investigated a single allocation algorithm (priority) and VDD-gating scheme (high water-mark) [13].

A. Costs of Gating Individual Registers

Fine-grained gating within a monolithic register file requires a PMOS-gate transistor with a control line and driver applied to each register. The PMOS-gate is only applied to the register bit-cells, as common circuitry (decoders, drivers, sense-amps etc.) cannot be VDD-gated without interfering with reads and writes to un-gated portions of the RF. While such an approach supports the finest granularity, its bit-cell limitation misses opportunities for leakage reduction. Table II shows the leakage contributions of each component in our baseline 160x64-bit register file, described in Table I. While the SRAM bit-cells contribute 28% of the leakage current, it is the shared circuitry that yields the greatest potential for energy reduction. A single PMOS of the same width as in the bit-cells is sufficient to gate all 64-bits in the register [5], yielding a 0.3% increase in total transistor width, with a 25% reduction in leakage energy-per-cycle. The PMOS-gate is driven by an inverter sized for a single FO4-delay. In our 45nm technology, the bit-cells must be disabled for 15-cycles to recoup the VDD switching cost of the PMOS and driver, shown in Table III.

B. Banked Register File Gating Costs

Register files are often partitioned to isolate shared RF circuits among the bank, allowing sub-banks to be clock-gated [8]. However this partitioning exacerbates leakage power consumption, as the number of bit-cells remains constant, but the relative amount of peripheral circuitry increases. Coarse-grained RF VDD-gating can provide larger leakage-energy reduction than gating bit-cells in a monolithic RF, but the opportunities to gate become more limited as granularity (RF bank size) increases. We investigate VDD-gating of an RF composed of banks of 4, 8, and 16 registers organized as shown in Figure 4. We isolate the output of each banks read-ports with a tri-state driver to prevent perturbations of the output [15].

The size of the PMOS gate, calculated using equation 1 [15], is determined by the maximum current through the bank and the amount of delay tolerated by the increased PMOS stack. We assume a delay increase (PGD) of 3% , where α (velocity saturation index coefficient) is calculated to be 1.27 via simulation, and R_m, V_{dd}, and V_t are library parameters.

$$W_{PMOS} = \frac{1}{1 - \sqrt[1-\alpha]{PGD}} \left(\frac{R_m}{V_{dd} - V_t} \right) \times I_{on} \qquad (1)$$

As before, there is a switching cost for toggling the bank and including isolation hardware vs. simply clock-gating a bank.The overhead and break-even of vdd-gating is summarized in Table III. We consider two PMOS sizes with delays of 1 and 5 cycles to reach VDD after enabling the bank. In both cases, the break-even point is consistent, as the driver overhead is proportional to the PMOS gate width. However, the smaller PMOS gate "costs" less in absolute amount of energy. This slower PMOS still meets our RF latency requirements.

We examine how VDD-gating and RF bank size affects RF power by looking at a typical case where a free-list is used to allocate registers. If a bank (or register) is empty, we assume it can be disabled immediately. In Figure 5, we show the average percent of the RF banks that are unallocated

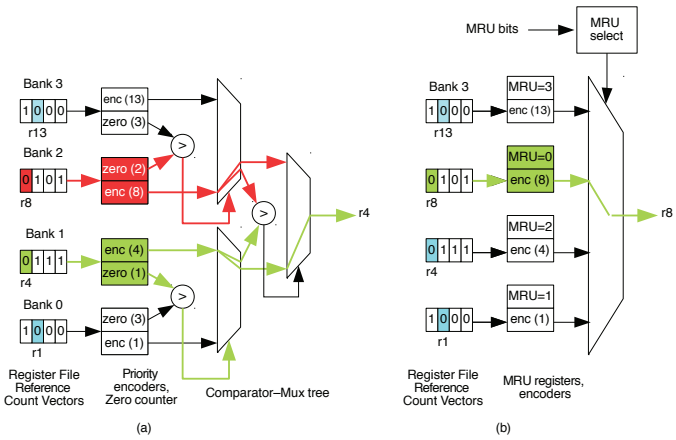

Fig. 7. **(a) Fullness Allocation:** Zero-counter blocks drive a comparator-mux tree propagating the lowest nonzero count and the corresponding register signifier. Bank1 is the fullest bank with only 1 free register, R4 is propagated as the next allocation. **(b) MRU Allocation:** MRU registers keep track of position in MRU stack. The MRU RF bank with an available slot is selected.

and able to be disabled. It is immediately apparent why previous work focused on gating of individual registers. Free-list allocation is not conducive to VDD-gating a partitioned RF, as bank-occupancy is too high. Increasing bank granularity only exacerbates this problem. Partitioned register files also have a larger static power cost due to duplicated periphery circuits. This cost is unrecoverable if free-list allocation is used, as there is limited opportunity to disable banks.

On the other hand, partitioning reduces RF dynamic costs. In Figure 6, we see the dynamic cost of clocking, reading, and writing the RF is much less for partitioned register banks. Partitioned banks contain fewer registers, leading to narrower decoders with significantly lower wire-delay allowing for smaller drivers. In the fine-grained case, we see that the total cost is reduced when register-pressure is low as leakage current is reduced. In the coarse-grain cases, energy per cycle is consistent as RF banks are rarely disabled.

V. RF Allocation and Gating Algorithms

In this section we describe algorithms that leverage microarchitecture information and investigate both *where* to place registers and *when* to toggle RF banks in order to maximize energy reduction. While we evaluate these algorithms with VDD-gating, they may also be used with drowsy and retention based schemes [8] where it is also important to know which registers are in use.

A. Allocation Algorithms

We evaluate two existing register allocation algorithms: free-list and priority based encoding using reference counts, along with three new schemes: fullness, most-recently used, and partitioned long-latency allocation.

Free List. The baseline allocation algorithm uses a MIPSR10k style circular-queue FIFO to manage allocation. Registers enqueue to free-list tail at commit and are dequeued from the free-list head when allocated to a dispatching instruction. The costs associated are an n-entry free-list FIFO.

Priority. Registers are allocated to the first free register in the RF. Instead of a free-list, this scheme uses reference counts for register management. The bit-vector representation of the RF indicates the status of each register: '1' indicates allocated, while '0' indicates free. A priority encoder reads the vector and outputs the first '0' register reference. The overheads include the reference-count vector, decoders, and priority encoders, and are of the same order as the free-list [13].

Fullness. This novel scheme modifies the priority approach by selecting the first available register in the *fullest* bank. An implementation is shown in Figure 7 where the number of zeroes in each bank's register reference count is compared. The lowest non-zero count and register signifier propagate through $log_2(n)$ mux stages. This mux-comparator tree is an additional cost over the priority scheme; however we can use significantly smaller encoders (from $0.1\times$ to $0.4\times$ as wide)

Most Recently Used (MRU). Registers banks keep a chronological history of each allocation, grouping younger instruction together by selecting the MRU bank. If the MRU bank is full, the most-recent bank with space is found. When a bank is selected, its MRU register is cleared and every other banks MRU register is incremented. The priority encoded value of the banks register reference count is selected via a $\frac{num.\ Pregs}{num.\ Ways}$ wide mux, illustrated in Figure 7 (b).

Long Latency. A portion of the RF is reserved for load operations. A load experiencing a cache miss will have significant latency, keeping its allocated register idle until the miss returns. Isolating loads should prevent registers allocated to these instructions from keeping the other RF banks enabled.

B. VDD-Gating Algorithms

The goal of a VDD-gating algorithm is to maximize both the number of banks that are disabled and the number of cycles that a bank is disabled. Toggling is to be avoided, as it will cause banks to be enabled prior to reaching the break-even point, thus costing more energy than it saves.

Immediate. Our baseline algorithm disables the bank *as soon as possible*. Once the bank is empty, the gating signal will be asserted. This has maximum opportunity for gating banks, but also maximum chances for unnecessarily toggling, as banks could be enabled immediately after being disabled. This algorithm couples well with *fullness* allocation, as an empty bank is only power on when every other bank is full.

Watermark-8. This algorithm keeps track of the number of active banks over the previous 8 cycles. The high watermark out of 8 counters is recorded, and all enabled but empty banks in excess are disabled. This conservatively tracks the register usage and should reduce toggling at the cost of missing opportunities to gate more banks. If insufficient banks are enabled, banks are enabled on-demand at register rename [13].

ROB %. This algorithm enables banks in proportion to ROB occupancy. As ILP increases, more register banks are enabled, while when the ROB entries are squashed or committed, banks are disabled if they are empty. Once the ROB is greater than 95% full, all empty banks are disabled as this indicates that a stall condition could occur, due to ROB

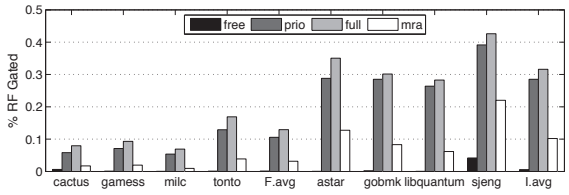

Fig. 8. **% RF Gated.** Banks of 4 regs, gated immediately when empty. Allocation algorithms are swept.

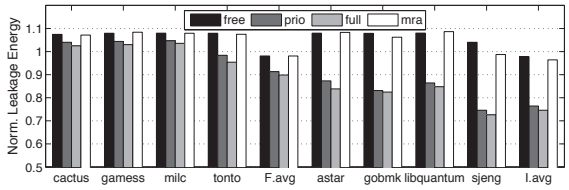

Fig. 9. **Leakage Energy** normalized to clk-gating empty banks. Banks of 4 registers sweeping allocation algorithms. (lower is better)

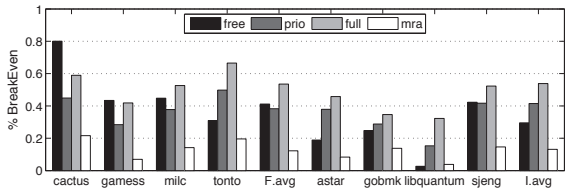

Fig. 10. **% VDD-toggles that break even** varying allocation algorithms (Higher is better). RF is configured with banks of 4 regs.

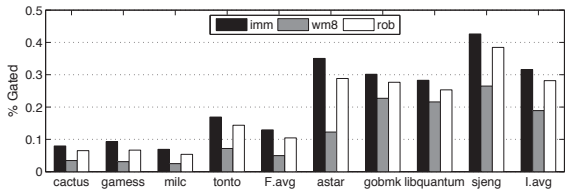

Fig. 11. **% RF Gated.** Configured with banks of 4 regs allocated using 'fullest' scheme. Gating algorithms are swept. (Higher is better)

starvation. An RF bank is re-enabled at rename if there are insufficient resources available.

C. VDD-Enabling Algorithms

We investigate two schemes for re-enabling RF banks:

Immediate. The baseline approach enables banks at rename when a register is allocated from that bank. This prevents starvation of resources as banks are enabled on-demand, but has the highest energy cost.

Delayed. This approach delays enabling the bank until 5-cycles after a register has been allocated from it. This is within the minimum allocate-write distance observed from Figure 2. We use a smaller PMOS VDD-gate transistor (shown in Table III to consume this slack.

VI. Experiments and Analysis

A. Allocation Experiments

We first investigate register-allocation by modeling a banked RF with banks of 4-registers that are gated immediately when the bank is empty. The register allocation algorithm is swept from free-list, priority, fullness, and most-recent.

RF Gating. Figure 8 shows the average percentage of the RF that is disabled during several SPEC benchmarks with aggregate date in columns F.avg and I.avg. As expected, the conventional free-list approach performs poorly across all workloads, due to the 'scattering' effect of the circular queue. *Most-Recent* performs similarly poorly; registers become scattered as the most-recent banks fill up. The *priority*-encoded scheme performs well, gating 10.5% and 28.5% of the RF for FP and INT workloads.

Fullness performs best overall, disabling 12.9% of the RF during FP workloads and 31.6% of the RF during INT workloads, an average improvement of 16.6% over the *priority* scheme. This improves upon *priority* by eliminating cases where allocation would re-enable an empty bank because it contains the *first* empty register. *Fullness* reduces the average energy cost of the partitioned RF by 30% vs. *free-list* allocation. Disabling more register banks reduces both the static power costs (by reducing the leakage current of disabled banks) and the dynamic costs (disabled banks are not accessed or clocked). Compared to free-list allocation, *fullness* reduces the dynamic RF energy cost from 31 to 23 $\frac{pJ}{cycle}$ for INT workloads, and from 32 to 29 $\frac{pJ}{cycle}$ for FP workloads.

Leakage Reduction. Figure 9 shows the RF leakage energy under each allocation scheme normalized to a banked-RF where empty banks are clock-gated instead of VDD-gated. *Most-recent* and *free-list* have the lowest savings due to poor register distribution, while *priority* and *full* perform significantly better. For workloads such as *F.cactus*, register pressure is sufficiently high that banks cannot be disabled long enough to improve over clock-gating in *any* allocation algorithm. Aggregating across all workloads (F.avg and I.avg columns) shows a benefit of up to 12% and 26% across all FP and INT workloads for *Fullness*.

Figure 10 gives further insight into why *free-list* and *most-recent* do not perform well, and why *full* performs better than *priority*. This figure shows the percentage of bank VDD-toggles that remain gated in excess of the toggling 'break-even distance', shown in Table III. Banks that are disabled for a period shorter than this distance cost energy, while banks disabled in excess of this save energy. *Fullness* performs significantly well across all benchmarks, with 35% more toggles breaking-even than *priority* due to built-in histeresis.

B. PMOS-Gating and Enabling Experiments

We investigated how gating algorithms affect RF performance by keeping bank size (4) and the allocation algorithm (fullness) constant. Figure 11 shows how varying the gating algorithm affects the amount of the RF that is enabled. *Immediate* performs best in this case, as banks are disabled once their reference count is empty. The disabled bank has the lowest priority to be re-enabled as all active banks are more full and will be preferred. *WM8* has the highest amount enabled as its watermark approach is slower to track changes in program behavior. The *ROB-proportional* approach tracks

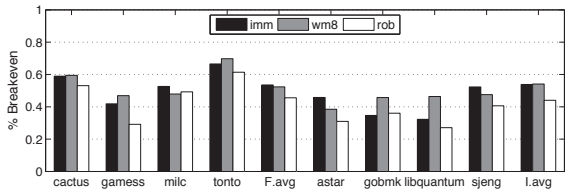

Fig. 12. **% VDD-toggles that break even** varying gating algorithms. RF configured with banks of 4 regs and 'fullest' allocation.

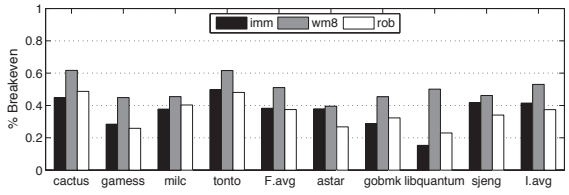

Fig. 13. **% VDD-toggles that break even** varying gating algorithms. RF configured with banks of 4 regs and 'prio' allocation.

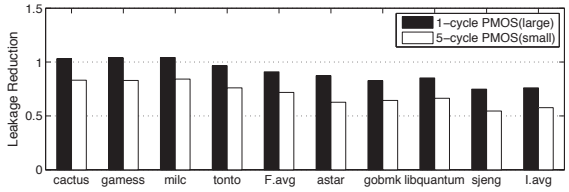

Fig. 14. **Leakage savings** vs clock gating banks (bank size=4) when varying PMOS gate-size, regs allocated to fullest banks

well with the *immediate* algorithm as ROB pressure acts as a proxy for register pressure.

Break-even. Figures 12 and 13 show how gating algorithms perform differently according to the allocation algorithm. Figure 12 shows the break-even percentage when *fullness* allocation is used. Fullness is relatively insensitive to the gating algorithm, with a slight preference to gating immediately as empty banks are de-prioritized. The *ROB-proportional* approach is too conservative and leaves banks enabled. Figure 13 shows the same experiments modeling a RF with *priority*-encoded allocation. In this case, *immediate* performs poorly due to a lack of hysteresis, while *WM8* performs better as it keeps a buffer of banks enabled.

VDD Latency. The minimum delay between when a bank is enabled due to a register allocation and when that register needs to be powered-on to receive data is 6 cycles, for our example processor. We take advantage of this by using a smaller PMOS gate that reaches VDD in 5 cycles rather than in 1 cycle. This arrangement has a reduced "toggle" cost and banks stay gated longer. While this has a negligible effect on dynamic energy, it reduces leakage considerably.

Figure 14 shows the leakage reduction when VDD-gating is applied using both large (1-cycle) and small (5-cycle) PMOS-gates compared to clock-gating RF banks. Now, even floating point workloads that previously preferred clock-gating now show benefits from VDD-gating. The smaller PMOS delays when switch-on occurs, significantly improving the number of toggles that break-even. The break-even ratio increases by 24% for both FP and INT workloads as more slack is absorbed, reducing leakage energy-costs by 22% for a RF with banks of 4 regs using *fullness* allocation. While this improvement comes with a performance cost increasing the RF read and write delay, the new cycle-time does not exceed our core 2-cycle access-latency requirement.

Partitioned. Registers allocated to load instructions that incur a cache-miss will remain allocated, but unused for 100's of cycles. Such long-latency instructions waste energy by preventing otherwise empty banks from being disabled. The RF is partitioned two sections with one reserved for load instructions to isolate them from the pool of general purpose registers. The partition size is swept from 0 to 8 banks (20% of the RF). The net result (not shown) is a negligible change in the percent of the RF that is gated. This scheme only identifies the head of a potential long-latency dependency chain, but neglects dependent instructions who are also consuming RF resources. Identifying only loads that are *likely* to miss or have already missed and the rest of the dependency chain will be key to improving this scheme.

C. Monolithic vs. Banked

In this section, we compare a monolithic-RF with fine-grained VDD-gating of SRAM bit-cells (bank=1) against banked RF configurations using *fullness* and *free-list* allocations with *immediate* gating and *delayed* enable. We vary bank-size from 4- to 16-registers to investigate if we can recover the leakage overheads from RF banking, recalling that RF leakage represents up to 30% of core-leakage [2], [3].

% Gated. Figure 15 shows the average size of the gated portion of the RF for each benchmark. Fine-grained bit-cell gating is most successful at gating, disabling 24% of FP 40% of INT workloads, independent of the allocation scheme. As coarseness increases, free-list based VDD-gating breaks down, while our *fullness* approach is able to consume resource slack, gating 6% to 12% for FP and 24% to 34% for INT workloads.

Leakage. Figures 16 and 17 illustrate the leakage overheads associated with banked register files. Figure 16 shows the normalized leakage energy for each configuration. Where previously we normalized to the RF clock-gating these banks, in this case, we normalize to the baseline monolithic-RF without any VDD-gating circuitry applied to illustrate the banking costs. The leakage-energy cost of banking can reach up to 1.5× the baseline for a RF composed of 40 4-bank registers, due to repeated SRAM-periphery circuitry and a larger number of large PMOS VDD-gate drivers.

Bit-cell gating uses 0.62× as much leakage energy as the baseline on average. In the free-list side, there is negligible banking for coarse-grained banks, so leakage remains high. The *fullness* algorithm can recover some of the leakage energy cost of coarser banks. When applied to a RF composed of 10 banks of 16 registers, the RF will use 0.56× as much energy as the baseline RF, and consume 0.89× as much static energy as the fine-grained bit-cell gated RF and will use 25% less energy than if a free-list approach were used.

Dynamic. Similarly for dynamic energy, allocation and gating algorithms can recover energy that is otherwise spent by

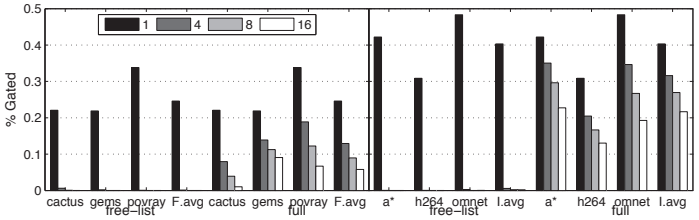

Fig. 15. **% RF Gated** for free-list and full allocations varying bank-size. Bank=1 indicates a monolithic RF with SRAM bit-cell gating.

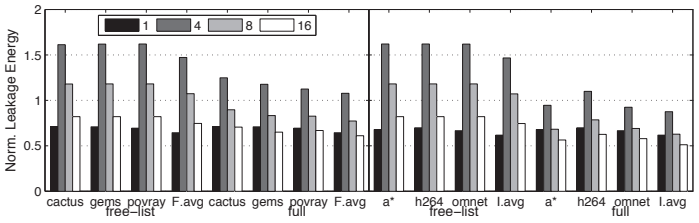

Fig. 16. **Leakage Energy/Cycle** for free-list and full allocations varying bank size. Normalized to baseline RF described in Table II

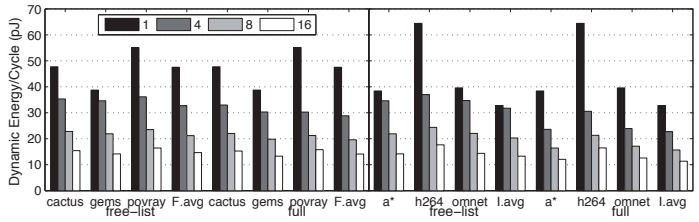

Fig. 17. **Dynamic Energy/Cycle** for free-list and full allocations sweeping bank-size. Bank=1 indicates a monolithic RF.

free-list allocation. Dynamic energy is reduced by partitioning the RF into banks as the cost for reading and writing a register is reduced up 5× lower in our 45nm technology. Even banked free-list allocation is cheaper than a monolithic-RF. Our *fullness* algorithm improves upon the free-list by opportunistically disabling RF banks, with energy reductions from 15% to 27% for INT workloads and 4% to 12% for FP workloads vs. the monolithic baseline, with larger dynamic savings from smaller banks.

VII. Conclusion

The distribution of registers across the register file is a critical determinant of VDD-gating efficacy. Limitations due to free-list FIFO allocation have caused previous works to focus on fine-grained gating of RF bit-cells, missing opportunities for larger energy reduction through RF partitioning. We use banking to both reduce the dynamic energy-cost of accessing the register file and to isolate circuits for larger leakage reductions. We investigated three new allocation algorithms (*full, latency-partitioned, mru*), compared against two existing schemes (*free-list, priority*) , and varied VDD-gating granularity from individual SRAM bit-cell to banks of 4- through 16-registers.

We incorporate detailed circuit models into our cycle-accurate simulation to measure the per-cycle cost of toggling and dynamically accessing RF banks. We investigate several algorithms to determine when to disable RF banks for maximum leakage reduction, with an *immediate* approach yielding best results when coupled with *fullness* allocation as it tracks bank occupancy. The allocation scheme provides hysteresis to prevent recently disabled banks from activating. A smaller PMOS gate is used to convert the minimum "allocate-use" distance into energy reduction, absorbing the pipeline slack. When applied to banks of 16-registers, these algorithms consume 0.76× as much static energy vs clock-gating the banked-RF instead. Compared to a monolithic bit-cell gated RF, fullness and immediate algorithms consume 0.89× as much static energy and 0.31× as much dynamic energy. With RF leakage occupying 30% of core power budgets, these savings can be critical for modern power-constrained cores.

VIII. Acknowledgment's

The authors thank the reviewers for their comments and Andrew Hilton from Duke University for his assistance with the x86 simulator. This work was supported by NSF grant CCF-1017184.

References

[1] D. Gonzales, "Micro-RISC Architecture for the Wireless Market," *Micro, IEEE*, vol. 19, no. 4, Jul-Aug 1999.

[2] V. Zyuban *et al.*, "Power Optimization Methodology for the IBM POWER7 Microprocessor," *IBM Journal of Research and Development*, vol. 55, no. 3, May-June 2011.

[3] E. Donkoh, T. S. Ong, Y. N. Too, and P. Chiang, "Register file write data gating techniques and break-even analysis model," in *Proc. of the Int. Symp. on Low Power Electronics and Design*, 2012.

[4] H. Esmaeilzadeh *et al.*, "Dark Silicon and the End of Multicore Scaling," in *Proc. of the Int. Symp. on Computer Arch.*, 2011.

[5] M. Powell *et al.*, "Gated-Vdd: a Circuit Technique to Reduce Leakage in Deep-submicron Cache Memories," in *Proc. of the 2000 Int. Symp. on Low Power Electronics and Design*, 2000.

[6] Z. Hu *et al.*, "Microarchitectural techniques for power gating of execution units," in *Proc. of the 2004 Int. Symp. on Low Power Electronics and Design*. New York, NY, USA: ACM, 2004.

[7] K. Flautner *et al.*, "Drowsy Caches: Simple Techniques for Reducing Leakage Power," in *Proc. of the Int. Symp. on Computer Arch.*, 2002.

[8] X. Guan and Y. Fei, "Reducing Power Consumption of Embedded Processors Through Register File Partitioning and Compiler Support," in *Proc. of the Int. Conf. on App.-Specific Sys., Arch. and Proc.*, 2008.

[9] R. Nalluri *et al.*, "Customization of Register File Banking Architecture for Low Power," in *Proc. of the Int. Conf. on VLSI Design*, 2007.

[10] J.-L. Cruz, A. González, M. Valero, and N. P. Topham, "Multiple-banked register file architectures," in *Proc. of the Int. Symp. on Computer Architecture*, 2000.

[11] D. Kanter, "Intel's Sandy Bridge MicroArchitecture." [Online]. Available: http://www.realworldtech.com/sandy-bridge/5/

[12] K. Yeager, "The MIPS R10000 Superscalar Microprocessor," *IEEE Micro*, Apr. 1996.

[13] S. Battle, A. D. Hilton, M. Hempstead, and A. Roth, "Flexible Register Management Using Reference Counting," in *Proc. of the Int. Symp. on High-Performance Computer Architecture*, 2012.

[14] N. K. Choudhary *et al.*, "Fabscalar: Composing synthesizable rtl designs of arbitrary cores within a canonical superscalar template," in *Proc. of the Int. Symp. on Computer Architecture*, 2011.

[15] Y. Shin, J. Seomun, K.-M. Choi, and T. Sakurai, "Power Gating: Circuits, Design Methodologies, and Best Practices for Standard-cell VLSI designs," *ACM Trans. Des. Autom. Electron. Syst.*, Oct. 2010.

[16] M. Goto and T. Sato, "Leakage Energy Reduction in Register Renaming," in *Proc. of the Int. Conf. on Dist. Computing Syst.*, 2004.

[17] S. T. Khasawneh and K. Ghose, "An adaptive technique for reducing leakage and dynamic power in register files and reorder buffers," in *Proc. of the Int. Conf. on Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation*, 2005.