Managing Leakage for Transient Data: Decay and Quasi-Static Memory Cells

Zhigang Hu[†], Philo Juang[†], Phil Diodato[‡], Stefanos Kaxiras[‡],
Kevin Skadron[*], Margaret Martonosi[†], Douglas W. Clark[†]
[†]Princeton Univ. [‡] Agere Systems [*]Univ. of Virginia
Princeton, NJ Allentown, PA Charlottesville, VA

ABSTRACT

Much of on-chip storage is devoted to transient, often short-lived, data. Despite this, virtually all on-chip array structures use six-transistor (6T) static RAM cells that store data indefinitely. In this paper we propose the use of quasi-static four-transistor (4T) RAM cells. Quasi-static 4T cells provide both energy and area savings. These cells have no connection to Vdd thus inherently provide decay functionality: values are refreshed upon access but discharge over time without use. This makes 4T cells uniquely well-suited for predictive structures like branch predictors and BTBs where data integrity is not essential. We use quantitative evaluations (both circuit-level and cycle-level) to explore the design space and quantify the opportunities. Overall, 4T based branch predictors offers 12-33% area savings and 60-80% leakage savings with minimal performance impact. More broadly, this paper presents a different view of how to support transient data in power-aware processors.

Categories and Subject Descriptors

B.7.1 [**Hardware**]: Integrated Circuits—*Types and Design Styles*

General Terms

Algorithms, Design, Measurement

Keywords

Leakage power, transient data, decay, quasi-static, memory cell

1. INTRODUCTION

As fabrication processes have worked to maintain clock speeds while scaling supply voltage, threshold voltages are being lowered to the point where leakage energy has become an important and growing fraction of total energy dissipation in high-performance CMOS CPUs. If it is not addressed through fabrication or circuit-level changes, some forecasts predict as much as a five-fold increase in leakage energy per technology generation [1]. At such rates, leakage energy would balloon to 50% or more of total chip energy in just a few generations.

Because large on-chip storage structures contain so many transistors, and because any transistor storing a charge leaks, most work in the architecture community on controlling leakage energy has focused on "turning off" portions of the on-chip arrays that appear not to be in use. Current on-chip arrays use six-transistor (6T) SRAM cells because they are fast and because they are truly static: a charge stored in a 6T cell will be maintained as long as that cell is connected to the drain and source voltages (V_{dd} and V_{ss}). Powell *et al.* [7] proposed a circuit technique called *gated-*V_{dd}, which disables a region of the cache by disconnecting it from V_{dd}. Kaxiras *et al.* [6] and Zhou *et al.* [12] describe architectural policies to guide *gated-*V_{dd}: individual cache lines which have not been used for a long time should be shut off because they tend to contain data that is not likely to be used again before replacement. These techniques use counters to gauge how long lines have been idle. Counter-based techniques are effective, but they have downsides as well. First, they have hardware overhead, albeit less than 5%. Second, their benefits are mainly limited to leakage energy.

This paper takes a different approach. Rather than devising new algorithms for identifying groups of transistors to turn off, we propose the use of a different memory cell: the *four-transistor* (4T) quasi-static RAM cell. 4T cells are about as fast as 6T cells, but they do not store charge indefinitely. Rather, the charge gradually leaks away at a rate that is a function of the cell's specific design as well as the current operating temperature. This caps the amount of leakage energy that an unused cell can dissipate. The area benefits of 4T cells stem from the fact that power and ground lines need not stripe vertically down the array.

4T cells are especially applicable to on-chip structures whose data is both transient and predictive. Transient, in the sense that data which has not been used for a sufficiently long time has decayed. And predictive, in the sense that allowing a value to leak away even if it will be used again does not harm correctness. Using a decayed value will possibly cause a misprediction that can be corrected by existing hardware. This is a key difference from caches, where using decayed data will lead to incorrect execution of the program.

Branch predictors are typical structures storing transient data. In this paper, we evaluate the effectiveness of branch predictors designed with 4T cells. We use quantitative evaluations (both circuit-level and cycle-level) of the branch predictors to measure the energy and area benefits. Overall, using 4T cells for branch predictors offers 60-80% leakage savings with minimal performance impact while providing an area advantage of 12-33%. More broadly, the paper begins a rethinking of how transient data should be supported in power-aware processors.

The rest of this paper is organized as follows. The next section provides background information about our simulations. Section 3 proposes a branch predictor design using 4T cells that automatically provides decay. Section 4 concludes the paper.

2. EXPERIMENTAL SETUP

2.1 Simulation Setup

Simulations in this paper are based on the SimpleScalar 3.0 tool set [3]. Our model processor has microarchitectural parameters that resembles in most respects the Intel PIII processor [4]. The main processor and memory hierarchy parameters are shown in Table 1. For performance estimates and behavioral statistics, we use SimpleScalar's *sim-outorder* simulator. For energy estimates, we use the Wattch simulator [2]. Wattch uses SimpleScalar's *sim-outorder* cycle-accurate model and adds cycle-by-cycle tracking of power

ISLPED'02, August 12-14, 2002, Monterey, California, USA.

Processor Core	
Instruction Window	40-RUU, 16-LSQ
Issue width	4 instructions per cycle
Functional Units	4 IntALU,1 IntMult/Div, 4 FPALU,1 FPMult/Div, 2 MemPorts
Memory Hierarchy	
L1 D-cache	16KB, 4way, 32B blocks, 3-cycle
L1 I-cache	16KB, 4way, 32B blocks, 3-cycle
L2	Unified, 256KB, 8-way LRU, 32B blocks,8-cycle latency, WB
Memory	100 cycles
TLB Size	128-entry, 30-cycle miss penalty
Branch Predictor	
Branch predictor	16K-entry gshare, 12 bits history
Branch target buffer	2048-entry, 4-way

Table 1: Configuration of simulated processor.

dissipation by estimating unit capacitances and activity factors.

We use spice-level tools from Celerity for detailed circuit simulations with a 25 pico-second resolution. The 6T and 4T RAM cells are taken from Agere Systems' cell libraries; no custom designs are assumed. 4T cells exist in these libraries because of their possible use as DRAM cells embedded onto a primarily-logic chip.

Process technology primarily determines leakage currents. We considered 3 Agere technologies shown in Table 2. With each successive generation, leakage increases exponentially. The leakage currents are shown for room temperature, 25C, and thus may underpredict the true leakage seen in running chips where the operating temperature is likely to be much higher. Figure 1 shows the leakage currents for varing temperatures for COM2 transistors. The exponential relation of leakage to temperature is evident in this figure. Our designs target an operational temperature of 85 C but we also discuss mechanisms to control 4T cells under very high temperature (125 C).

	COM2	COM3	COM4
Feature Size (um)	0.16	0.12	0.1
Vdd (V)	1.5	1.0	1.0
Transistor Leakage Current (nA)	3	10	100

Table 2: Comparison of Agere COM2, COM3 and COM4 technologies. Leakage currents are for 25°C.

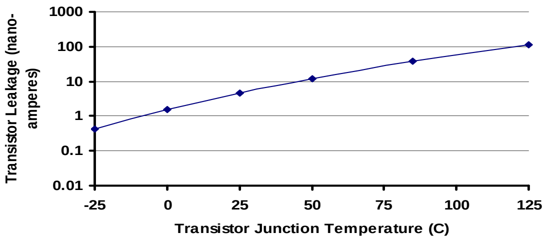

Figure 1: Transistor leakage current (nA) for varying temperatures for the COM2 process.

2.2 Benchmarks

We evaluate our results using benchmarks from the SPEC CPU2000 suite [9]. The benchmarks are compiled and statically linked for the Alpha instruction set using the Compaq Alpha compiler with SPEC *peak* settings and include all linked libraries. For each program, we skip the first 1 billion instructions to avoid unrepresentative behavior at the beginning of the program's execution. We then simulate 500M (committed) instructions using the reference input set. Simulation is conducted using SimpleScalar's EIO traces to ensure reproducible results for each benchmark across multiple simulations.

3. BRANCH PREDICTOR DESIGN WITH QUASI-STATIC 4T RAM CELLS

Quasi-static 4T memory cells are mainly considered as a means of implementing DRAM within a logic fabrication process [8, 10]. In traditional uses, the perceived drawback of the 4T cells is that they are dynamic and need refresh; but this characteristic is actually the key for an elegant decay design. In contrast to previous 6T leakage control strategies, we do not have to turn off power to 4T cells. Instead, we let inactive cells decay naturally, thus avoiding any overhead associated with turning power on and off. Because of their use as embedded DRAM in some designs, 4T cells are already present in many design libraries, including those used by Agere. We use the cells as they appear in the library.

In addition, branch predictor data are not architectural, meaning that if we unknowingly lose them only performance might suffer but not correctness. This leads to a clean design without any decay counter hardware. The drawback in accessing decayed data is a potential bad prediction. As long as this is a rare event we can eliminate all the decay counter hardware and get similar benefits as in a 6T decay predictor. 4T cells are also smaller than 6T cells, thus offering area advantages also.

3.1 The Quasi-Static 4T Cell

Basic 4T DRAM cells are well established and described in introductory VLSI textbooks [11]. 4T cells are similar to ordinary 6T cells but lack two transistors connected to Vdd that replenish the charge that is lost via leakage (Figure 2). Using exactly the same transistors as in an optimized 6T design the 4T cell requires only 2/3 of the area. 4T DRAM cells naturally decay over time (without the need to switch them off); once they lose their charge they leak very little since there is no connection to Vdd.

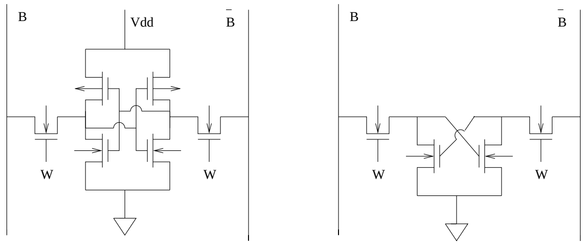

Figure 2: Circuit diagrams of the 6T SRAM cell (left) and the 4T quasi-static RAM cell (right).

Also importantly, 4T cells are automatically refreshed from the precharged bit lines whenever they are accessed. When a 4T cell is accessed, its internal high node is restored to high potential, refreshing the logical value stored in it; there is no need for a read-write cycle as in 1T DRAM. As the cell decays and leaks charge, the voltage difference of its internal nodes gradually drops to the point where the sense amplifiers cannot distinguish its logical value. Conservatively, this occurs when the node voltage differential drops below a threshold of the order of 100 mV (with 1.5V Vdd). Below this threshold we have a decayed state, where reading a 4T DRAM cell may produce a random value —not necessarily a zero. Over a long time the cell reaches a steady state where both the high node and the low node of the cell "float" at about 30mV.

4T cells possess two characteristics fitting for decay: they are refreshed upon access and decay over time if not accessed. In the rest of this section we discuss extensively the 4T decay design, including retention times and other considerations.

3.2 Retention Times In 4T Cells

We define *retention time* to be the time from the last access to the time when the internal differential voltage of the cell drops below the detection threshold. Retention time depends on the leakage currents present in the 4T cell. Retention time is a critical parameter for a 4T design because when implemented in 4T cells, decay techniques have the cell's retention time as their natural decay interval.

To study retention times for the 4T branch predictor we chose the Agere COM2 CMOS process for which we have accurate transistor models. Although our initial retention time numbers are for COM2, we feel they are accessible in COM3 and COM4 as well.

Retention time is affected by the characteristics of the transistors themselves. For example, doubling the channel length and the gate oxide thickness can extend the retention time by lowering leakage currents. In contrast to standard 4T transistors, we refer to these transistors as slow-decay transistors. The trade-off using slow-decay transistors is that the area advantage is reduced because RAM cells built upon these transistors are about 7/8 of the 6T cell.Table 3 compares the three cell types for their access time and cell area.

	4T		6T
	standard	slow-decay	
access time(ps)	525	565	490
RAM cell area(relative)	0.66	0.88	1

Table 3: Comparison of three cell types: standard 4T, slow-decay 4T and 6T cells

Variations in temperature also result in large variations in retention times. Our designs target an operational temperature of 85 C (appropriate for example for mobile processors) but we also discuss mechanisms to protect performance in situations where very high temperature (125 C) does not allow for sufficiently large retention times. Later in this section, we discuss methods for controlling retention times in 4T cells.

	standard 4T			slow-decay 4T		
	25C	85C	125C	25C	85C	125C
Ret.Time(ns)	18K	1.7K	0.56K	1M	57.2K	9.4K

Table 4: Retention times in nanoseconds for standard and slow-decay versions of 4T cells at different operating temperatures. For a 1GHz (1ns cycle time) processor, one can also consider these retention times as cycle counts.

Based on these assumptions, we determined retention times for our technology through detailed transistor-level simulations. We simulated an access to a cell, followed by a long period in which the cell was left unread. During this time, leakage causes the cell's internal nodes to lose charge. Recall that the retention time is the duration between an access and the point at which the differential voltages of the 4T cells internal nodes lapsed to a value less than 100mV. We used 100mV as our criteria for the minimum voltage we would expect the sense amplifiers to distinguish. Reading a decayed cell produces a valid, though random, predictor value. (We model this randomness in our simulated results that follow, and we discuss the finer points of this issue later in this section.) Table 4 gives the cell retention times in nanoseconds for the COM2 technology.

3.3 Locality Considerations

Granularity is also relevant in the 4T design but here it stems from the way 4T cells are refreshed. Branch predictors are typically layout as a square, with each row having multiple neighboring predictors. In a squarified predictor, reading a row refreshes all the cells in a row because the wordline is asserted. (Segmented wordlines would allow more selective refresh but these designs are outside the scope of this paper.)

Retention time selection and locality granularity go together because large row granularity make the apparent rate of refresh much higher. Cells that would have decayed if left alone get refreshed coincidentally by nearby active cells. Thus 4T cells with short retention times may not lose data as quickly if the row size is long enough. In contrast, in a design with very fine row granularity one would opt for 4T cells with very long retention times. Fine granularity leads to a very good decay ratio but the important cells must remain alive on their own (without the benefit of accidental refreshes) for considerable time.

3.4 Results for 4T-based Branch Predictors

We now examine the leakage and performance impact of branch predictor decay based on 4T structures. We considered a range of technologies, for this section, including COM2, COM3, and COM4. COM2 shows modest improvements with careful design, and future technologies improve significantly on this. We use slow-decay 4T cells in our design, both in the BTB and in the direction predictor. As for the overall configuration, we use a 16K-entry gshare configuration as in Table 1. We target an operational temperature of 85 C; this leaves us a decay interval of 57,200 cycles.

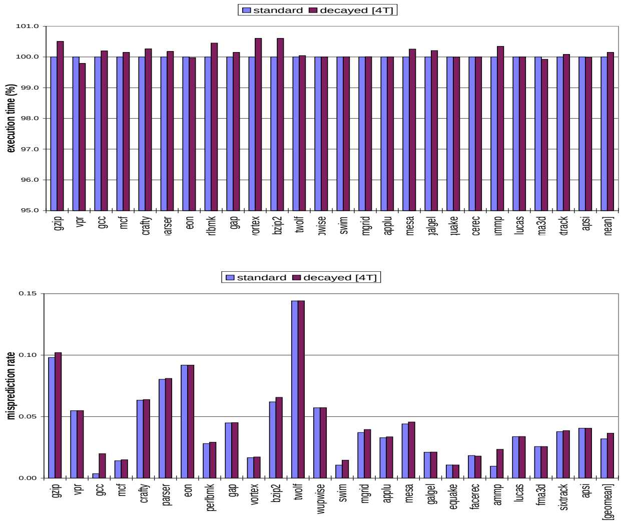

Figure 3: Normalized execution time (Top) and misprediction rate (Bottom) of standard and 4T predictors. 4T predictors produce minimal performance losses.

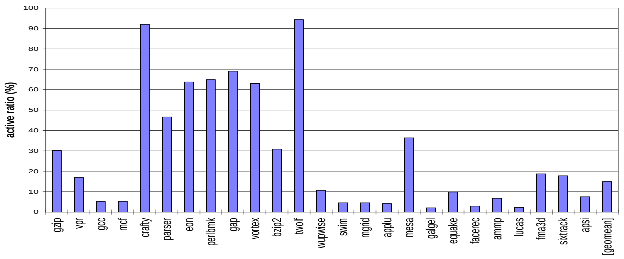

Figure 4: Active ratio of a 4T based predictor.

Figure 3 (top) shows the normalized execution time (in percentages) comparing conventional non-decaying 6T branch predictors and 4T based branch predictors. Note that the y-axis of the graph has a very limited range. From the graph, we see that execution time is virtually unchanged. That is, the performance impact of predicting branches based on decayed predictor entries is negligible. In fact, a few benchmarks actually improve slightly due to the random effects of reading decayed values. Furthermore, prediction accuracy (Figure 3, bottom) was also virtually unchanged. Over all the benchmarks, the overall prediction accuracy was down less than 0.5%. Figure 4 shows the active ratio of the direction counters. On average, we see a 15% active ratio, which directly translates into over 85% savings on leakage power over a traditional, non-decaying predictor.

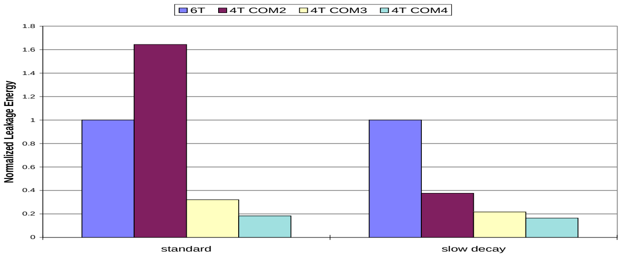

Figure 5: Normalized leakage energy for branch predictors with standard (Left) and slow-decay (Right) 4T cells.

Finally, the normal dynamic energy overhead of additional mispredictions must be included in our results. Using a calculation similar to that found in [6], we can evaluate the impact of additional dynamic overhead caused by decayed (and possibly mispredicted) reads. Note that this number is an energy calculation for the *entire* processor; that is, the dynamic overhead is the extra energy expended by the whole processor due to a longer runtime.

Figure 5 shows the normalized leakage energy with 4T based branch predictors. The leakage energy of a 6T branch predictor is defined as 1; a number lower than that indicates a processor equipped with a particular branch predictor consumed less energy, and vice versa.

As shown in the plot, we see that a processor with a branch predictor using either the standard 4T (Figure 5, Left) or slow decaying 4T (Figure 5, Right) cells consumes less energy under the COM3 and COM4 processes. At COM2, the branch predictor is decaying so rapidly that a lot of useful information is being discarded, imposing a performance penalty so severe that the overall energy consumed by the processor actually increases.

More importantly, we see the impending concern over leakage power more clearly; at COM3 and COM4, where leakage energy has a much larger impact, we can very aggressively decay using standard 4T cells and still achieve an overall power savings.

Examining the BTB decay reveals similar observations. Because each BTB target is much larger than the two-bit counter, we can afford to attach counters to each BTB target and thus achieve a very low active ratio.

Overall, we see that 4T cells provide immense leakage power savings with a minimal performance impact. We also see that overall, the processor will consume less energy despite the performance overhead, and that as leakage energy increases in influence vis-a-vis dynamic energy, a 4T based branch predictor becomes much more effective.

3.5 Discussion

This section expands on some key additional issues regarding branch predictors based on 4T cells.

3.5.1 Controlling Retention Times

The success of a 4T design depends on matching retention times to access (i.e., “refresh”) intervals. Thus, the ability to control retention times could give us a new degree of freedom in designing 4T structures.

A way to affect retention times is to add devices such as resistors or capacitors to the basic 4T cell [5]. Such devices can be used to slowly replenish the lost charge. If the rate of replenishment is less than the leakage the cell will still decay albeit much more slowly, thus retention time can be extended significantly.

3.5.2 Metastability

Another key issue regarding 4T structures is the fact that metastability problems are possible when the cell’s internal differential voltage is too small. To avoid metastability, it is tempting to use refresh, but this would obviate the savings of our approach. Instead, one can detect the small differential voltages and avoid relying on array data at these points. As an example of the latter, we propose that one could avoid metastability in a 4T branch predictor by adding a reference column whose sole purpose is to detect low differential voltage and to prevent the sense amplifier output from propagating further into the circuit. In this column, instead of a sense amplifier we have a voltage comparator. When the voltage difference is too small, the comparator output forces the reference cell to read as a logical zero (otherwise it reads as a logical one). The output of the comparator qualifies the result. A small differential is therefore prevented from inducing metastability in the subsequent circuit.

Other approaches are possible, such as the use of decay counters [6], but the one we have proposed—the use of a single comparator in a reference column—is appealing because it prevents metastability while requiring minimal extra area or power.

4. CONCLUSIONS

In this paper, we examine the use of quasi-static 4T cells for implementing branch predictors. Such 4T cells have been proposed to implement on-chip embedded DRAM in a fairly-traditional style with refresh circuitry. In our work, we examine using the natural decay of the 4T cells to implement decay for leakage-control in branch predictors. Because branch predictors are performance hints, not correctness-critical, lost entries do not cause incorrect execution. Moreover, we show that 4T cells can be built with sufficient natural retention times to implement useful decay-based predictors with negligible impact on prediction rate.

While branch predictor leakage is up to 10% of total CPU leakage, we are able to reduce it by a significant fraction, sometimes 90% or more. This reduces overall chip leakage by 5-7%. Furthermore we can reduce leakage with essentially no performance cost and in general, an area improvement of 12%-33%. Most broadly,the paper prompts a rethinking of how transient data can best be exploited in designing power-efficient processors.

5. REFERENCES

[1] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4), 1999.

[2] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architecture-Level Power Analysis and Optimizations. In *Proc. ISCA-27*, ISCA 2000.

[3] D. C. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. *Computer Architecture News*, 25(3):13–25, June 1997.

[4] K. Diefendorff. Pentium III = Pentium II + SSE. *Microprocessor Report*, Mar. 8 1999.

[5] P. Diodato et al. Embedded dram: An element and circuit evaluation. In *IEEE Custom Integrated Circuits Conference*, Jun 2001.

[6] S. Kaxiras, Z. Hu, and M. Martonosi. Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power. In *Proc. ISCA-28*, July 2001.

[7] M. D. Powell et al. Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories. In *ISLPED*, 2000.

[8] S. Schuster, L. Terman, and R. Franch. A 4-device cmos static ram cell using sub-threshold conduction. In *Symposium on VLSI Technology, Systems, and Applications*, 1987.

[9] The Standard Performance Evaluation Corporation. WWW Site. http://www.spec.org, Dec. 2000.

[10] A. G. Varadi. Quasi-static mos memory array with standby operation. US Patent Number 4,120,047.

[11] W. Wolf. *Modern VLSI Design: Systems on Silicon*. 1998. Prentice-Hall.

[12] H. Zhou, M. Toburen, E. Rotenberg, and T. Conte. Adaptive mode control: A static-power-efficient cache design. In *Proc. PACT2001*, Sept. 2001.