Leakage Energy Reduction in Value Predictors through Static Decay

Juan M. Cebrián, Juan L. Aragón and José M. García

Dept. Ingeniería y Tecnología de Computadores
Universidad de Murcia
30100 Murcia, Spain
{jcebrian, jlaragon, jmgarcia}@ditec.um.es

Abstract

As process technology advances toward deep submicron (below 90nm), static power becomes a new challenge to address for energy-efficient high performance processors, especially for large on-chip array structures such as caches and prediction tables. Value Prediction appeared as an effective way of increasing processor performance by overcoming data dependences, but at the risk of becoming a thermal hot spot due to the additional power dissipation.

This paper proposes the design of low-leakage Value Predictors by applying static decay techniques in order to disable unused entries from the prediction tables. We explore decay strategies suited for the three most common Value Predictors (STP, FCM and DFCM) studying the particular tradeoffs for these prediction structures. Our mechanism reduces VP leakage energy efficiently without compromising VP accuracy nor processor performance. Results show average leakage energy reductions of 52%, 65% and 75% for the STP, DFCM and FCM Value Predictors, respectively.

1. Introduction

Power dissipation and energy consumption are two major goals when facing the design of a new processor in the high performance computing domain –servers and high-end computing systems–. High-end systems, current teraflop computers and future petaflop computers, use many power-hungry components to achieve the required high performance. However, the power needs of these systems must be limited in order to make them feasible, trying not to reach intolerable operating costs (e.g., the Earth Simulator supercomputer requires 18 megawatts of power). Other power-related parameters such as leakage, temperature, failure rates, or component's life expectancy are also crucial for the success of high-end computing systems.

While feature size shrinks to allow greater density of transistors on the processor, supply voltage must be lowered to restrain dynamic power consumption since it is proportional to the square of the supply voltage. By doing this, however, static power is not reduced. This was not a problem for several generations, as leakage was only a minimal part of the overall power consumption and it was not considered as a major concern. But, when using smaller geometries (below 65nm) with very small threshold voltages, leakage loss is exponentially increased, to a point where static power begins to dominate the overall power consumption [11].

In order to deal with this problem, several proposals can be found in the literature at both circuit and architecture level for managing leakage power. Most of these proposals have focused on reducing the leakage power by switching off unused portions of large array structures. Cache Decay [10] selectively turns individual data cache lines off if they have not been used in a long time, reducing leakage power at the cost of loosing the content of the cache line. This *non-state preserving* technique has also been successfully applied to branch predictors [6][9].

Value Prediction (VP) has also been proposed [4][5][8][13] as an effective way of improving superscalar processor performance by overcoming data dependences. However, the use of VP structures, despite the speedup provided in superscalar processors (average speedup of 15% as reported in [2]), has not been widely spread, mainly due to complexity-delay issues.

In addition, the use of VP structures incurs in extra dynamic and static power dissipation. As in the case of caches and branch predictors, the continuous access to the VP structure may result in a thermal hot spot. Although the VP is a small structure compared to a L2 cache, if we let it overheat (likely, as it is accessed frequently and resides quite close to the core) without any precaution to

1-4244-0910-1/07/$20.00 ©2007 IEEE

regulate its leakage, the negative effects can be quite serious.

Regarding the complexity-delay issues related to VPs, note that, unlike other prediction structures such as branch predictors where increasing access time and complexity can significantly reduce their benefits (the next fetch instruction is needed as early as possible), the access time in Value Predictors is not so crucial. First, the predicted value is not needed until the instruction has reached its issue stage, and second, current high performance processors typically implement deep pipelines (14 stages or more) which effectively *hide* the VP latency due to the increased front-end pipeline length.

In this paper we propose static *Value Prediction Decay*, a mechanism able to dramatically reduce the leakage energy of traditional Value Predictors with negligible impact on accuracy and processor performance, especially for deep-submicron designs (below 90nm.) by locating and disabling unused entries of the predictor. Our proposal makes Value Predictors complexity-effective structures (due to the minimal extra hardware required) when used in medium and long pipelines as well as a power-performance efficient mechanism suitable for power-aware high performance computing systems.

The rest of the paper is organized as follows. Section 2 reviews some related works and provides an overview on Value Prediction. Section 3 analyzes the dynamic utilization of the prediction tables. The proposed *Value Prediction Decay* mechanism is described in Section 4. Section 5 shows the experimental methodology and the leakage energy savings obtained. Finally, Section 6 summarizes the main conclusions of the work.

2. Related work

In order to reduce leakage power in processors, several techniques have been proposed at both circuit and architectural level. At the architectural level, many proposals have focused on reducing the leakage power by switching off unused portions of large array structures such as caches. These techniques have been categorized into *state-preserving* and *non-state preserving* [1][7][14].

Studies by Powell *et al.* [12] proposed *gated-V_{DD}* as a technique to limit static leakage power by banking and providing “sleep” transistors which dramatically reduce leakage current by gating off the supply voltage. This technique, known as *decay,* reduces the leakage power drastically at the expense of loosing the cell’s contents, being necessary to apply it very carefully since the loose of information can result in an increase of the dynamic power to retrieve it again. Kaxiras *et al.* [10] successfully applied decay techniques to individual cache lines in order to reduce leakage in cache structures (67% of static power consumption can be saved with minimal performance loss). This technique has also been applied to conditional branch predictors and BTB structures [6][9].

On the other hand, *drowsy* techniques try to reduce leakage without loosing the cell’s information. Drowsy caches [3] use different supply voltages according to the state of each cache line. The lines in drowsy mode use a low-voltage level, retaining the data, while requiring a high voltage level to access it again. Waking up from the drowsy state is similar to a pseudo-cache miss incurring in some additional penalty cycles (about 7 cycles).

Li *et al.* [7] evaluated the use of state and non-state preserving techniques in caches. The authors showed that for a fast L2 cache (5-8 cycles latency) decay techniques are superior in terms of both performance loss and energy savings to drowsy ones.

Finally, an alternative to traditional decay is to use quasi-static, four-transistor (4T) memory cells. 4T cells are approximately as fast as 6T SRAM cells, but do not have connections to the supply voltage (V_{SS}). Rather, the 4T cell is charged upon each access, whether read or write, and it slowly leaks the charge over time until, eventually, the value stored is lost. In [9], it was proposed to apply decay techniques to branch predictors by using 4T cells. By doing this, some of the drawbacks of using *gated-V_{DD}* transistors are eliminated, since an access to a 4T cell automatically reactivates the cell, whereas reactivating a 6T cell from the “sleep” mode is somewhat more complex, requiring extra hardware involved in gating the supply voltage.

2.1. Value Prediction overview

The *last value predictor* was introduced by Lipasti *et al.* [8]. This is the most basic prediction mechanism and, basically, it assumes that the next value produced by an instruction will be the same as the previous one.

A generalization of the last value predictor leads to the *stride value predictor* (STP). Introduced by Gabbay *et al.* [4], it uses the last value produced by an instruction plus a stride pattern. The next predicted value is computed by adding the last value to the stride.

The *finite context method* value predictor (FCM), introduced by Sazeides *et al.* [13], uses the history of recent values, called the *context*, to determine the next value. This is implemented by using two-level prediction tables.

The *differential finite context method* value predictor (DFCM), introduced by Goeman *et al.* [5], joins the two previous predictors. DFCM works like FCM (two-level prediction tables), but it stores the *differences* between the values instead of the values themselves, plus the last value of the instruction. This allows DFCM to capture both stride and non-stride patterns.

3. Utilization analysis of Value Prediction structures

The power dissipation of Value Prediction structures is divided into dynamic and static power. The dynamic component depends on the utilization of the VP. Values can be predicted at different levels, the most aggressive utilization predicts the output value for all instructions traversing the pipeline. This is the worst case scenario for our mechanism, and it will be used in all our simulations. Other approaches, however, restrict the use of the VP to just a fraction of instructions such as long-latency instructions; instructions that belong to a critical path; or just to predict the effective address to provide memory disambiguation.

Therefore, limiting the VP utilization to just a fraction of selected instructions, effectively reduces the dynamic power component of this structure. However, the static power component is still present, as the VP structure leaks regardless of utilization with increasing leakage loss as process technology shrinks. For this reason, this work is focused on reducing the static power component of the VP structure.

In [10] Kaxiras *et al.* showed that, very frequently, cache lines have an initial active period (known as *live time*) followed by a period of no utilization (known as *dead time*) before they are eventually evicted. They proposed to break the stream of references to a particular cache line into generations. Each generation lasts until the cache line is evicted and replaced by a new one. This generational behavior also appears in Value Predictors, although with some particularities: as VPs are implemented as direct-mapped tables with no tags and allowing destructive interferences, in our proposal, a generation ends when the VP entry is accessed by an instruction with a different PC (see Figure 1). Its *live time* will be the period of accesses with the same PC and its *dead time* will be the period between the last access with an specific PC until an access with a new one.

Our first study analyzes the utilization of the VP tables, measuring the fraction of time each entry remains in the *dead* state, in order to determine if turning those VP entries off will result in a significant decrement of leakage energy. Figure 2 shows the average fraction of time each generation is in a *dead* state (i.e., the ratio *dead*/(*live*+*dead*)) for the whole SPECint2000 benchmark suite as a function of VP size (see Section 5 for details about simulation methodology and processor configuration).

It can be observed that the three evaluated Value Predictors –STP, FCM and DFCM– present a similar utilization regardless of their size. For sizes around 20 KB, the average fraction of dead time is 43% and for predictor sizes around 40 KB the average fraction of time the entries spend in their *dead* state is 47%. Therefore, if we were able to take advantage of these *dead times* by detecting them and shutting the entries off, we could reduce the leakage power of the Value Predictor structure by one half on average.

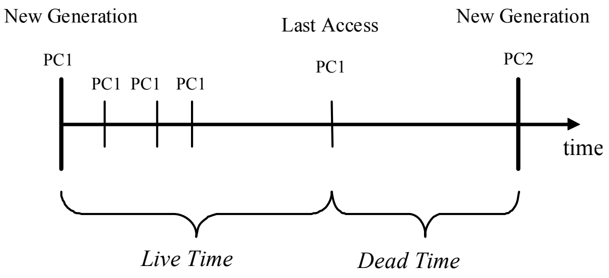

Figure 1. Generations in the value predictor.

However, it is important to note that this is not an upper bound on the leakage energy savings that could be achieved by decaying VP entries. Long periods of inactive *live time* could also be detected to early shut the entry off in order to obtain further leakage savings, at the expense of reducing the VP accuracy, as we will show in next sections.

4. *Value Prediction Decay* mechanism

In order to apply decay techniques to the Value Predictor mechanism, we need to detect VP entries that have been unused for a significant period of time in order to switch them off. Therefore, it is necessary to choose carefully the number of cycles we should wait before shutting an entry off to match generational changes.

The proposed *Value Prediction Decay* mechanism tracks the accesses to each VP entry in order to detect if a particular entry is accessed very frequently or, otherwise, the entry is unused for a long period of time, entering into a *dead* state. In order to measure the power-efficiency of our proposal, we will explore a wide range of *decay intervals* to precisely detect the *dead* states while, at the same time, not degrading the VP's accuracy.

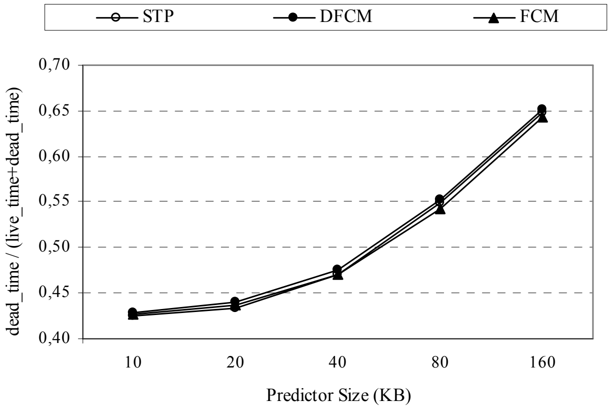

Figure 2. Fraction of time VP entries are in dead state (average SpecInt2000).

Note that, if the policy that decides when to turn a VP entry off uses too long decay intervals, the potential leakage savings will be reduced. Conversely, if the policy uses too short decay intervals, the VP hit ratio will be degraded. In any case, a positive effect of *Value Prediction Decay* when compared to *Cache Decay* is that prematurely disabling a VP entry is not so harmful as disabling a cache line: loosing the contents of the cache line *always* leads to an extra access to L2 cache or memory to retrieve the lost information incurring in extra execution cycles; however, loosing the contents of a VP entry might result –or not– in a value misprediction on the next access to that entry but this is exactly what would happen if we had a real generation change (which is a very common situation and one of the major limitations in traditional non-tagged VPs, where the huge number of destructive interferences dramatically shortens the generational replacement).

Regarding the utilization of Value Predictors, throughout the paper we are predicting the output values for *all* instructions traversing the pipeline. However, it is important to note that this aggressive prediction scheme does not benefit a decay mechanism. The more demanding use of the VP structure, the less opportunities to detect unused VP entries and the less leakage energy savings obtained from a decaying mechanism.

A power-efficient implementation of the decay mechanism requires the use of a hierarchical counter composed of a global counter and two-bit saturated gray-code counters on each VP entry[1]. Each time the global counter gets to zero, all the local counters will be incremented by one. On the other hand, an access to a VP entry results on a reset of its local counter. When a VP entry remains unused for a long time, its local counter will reach the upper limit eventually, and the corresponding entry will be shut off.

The length of the decay intervals is controlled by the period of the global counter. If we set the period of the global counter to a low value, the VP entries may be disabled prematurely and leakage will be reduced drastically, but so will be the hit rate of the predictor. On the other hand, if we disable too late (large global counter periods), the leakage energy savings won't be as high as they could have been.

The VP entries will be shut off by using *gated-V_{DD}* transistors [12]. These "sleep" transistors are inserted between the ground (or supply) and the cells of each VP entry, which reduces the leakage in several orders of magnitude and can be considered negligible. An alternative to using *gated-V_{DD}* transistors consists on using quasi-static 4T transistors in the VP array, although similar leakage savings would be expected [9].

Table 1. Configuration of the simulated Processor.

Processor Core	
Process Technology:	70 nanometers
Frequency:	5600 MHz
Instruction Window:	128 RUU, 64 LSQ
Decode Width:	8 inst/cycle
Issue Width:	8 inst/cycle
Functional Units:	8 Int Alu; 2 Int Mult 8 FP Alu; 2 FP Mult 2 Memports
Pipeline:	22 stages
Memory Hierarchy	
L1 Icache:	64KB, 2-way
L1 Dcache:	64KB, 2-way
L2 cache:	1MB, 4-way, unified

In order to precisely evaluate the net leakage energy savings provided by the static VP decay approach, it is necessary to consider the following overheads associated with the mechanism.

The first component overhead takes into account the extra dynamic and static power that results from the additional hardware (a global decay interval counter as well as the two-bit local counters per VP entry). The dynamic and static power overhead of the global counter and all 2-bit local counters has been measured to be less than 2% of the total VP structure for all evaluated sizes, which can be considered negligible.

The second overhead component is derived from the induced VP misses (when a VP entry is prematurely disabled) that increase execution time. These extra cycles that the program is running will also lead to additional static and dynamic power dissipation. Note that this second overhead is highly destructive since each extra cycle accounts for the overall processor dynamic and static power and it can easily cancel whatever VP leakage energy savings provided by the static decay scheme.

5. Experimental results

5.1. Simulation methodology

To evaluate the power-performance efficiency of the proposed *Value Prediction Decay*, we have used the SPECint2000 benchmark suite. All benchmarks were compiled with maximum optimizations (-O4 -fast) by the Compaq Alpha compiler and they were run to completion using the reduced input data set (test). The energy evaluation has been carried out with a modified version of the *HotLeakage* power-performance simulator [15] that includes the dynamic and static power model for the three evaluated Value Predictors (STP, FCM and DFCM) as

[1] Using a hierarchical counter is more power-efficient since it allows accessing the local counters at a much coarser level. Accessing the local counters each cycle would be prohibitive because of the power overhead.

well as both power overheads associated to our proposal. The access latency for the three Value Predictors is 5 cycles.

Table 1 shows the configuration of the simulated architecture. The leakage related parameters have been taken from the Alpha 21264 processor (provided with the *HotLeakage* simulator suite) using a process technology of 70 nanometers.

5.2. Leakage-efficiency of *Value Prediction Decay*

This section presents the power-performance evaluation of the proposed *Value Prediction Decay* mechanism for the STP, FCM and DFCM Value Predictors for different predictor sizes and for several decay interval windows: 64, 256, 512, 1024, 4096, 32768 and 262144 cycles.

For each evaluated Value Predictor we report, firstly, the IPC degradation as we reduce the decay interval, and secondly, the corresponding leakage energy[2] savings just for the VP structure. Overall leakage energy savings are not presented due to *HotLeakage* limitations that only provide static power models for regular array structures, such as caches and predictors, or the register file.

Figure 3 shows the performance degradation for the STP Value Predictor for different predictor sizes (averaging the whole SpecInt2000 suite). Note that, as cited previously, traditional VP (with no decay) can provide significant average speedups (12% for a 20 KB STP predictor). Looking into the performance degradation caused by *static decay*, we can notice that it is degraded as expected, due to the data loss of prematurely deactivating VP entries that still are in a *live state*. In particular, there is a slight IPC degradation (around 1%) for 1024- and 512-cycle decay intervals. However, due to early deactivation of entries and the induced extra execution cycles, for 256-cycle, and smaller, decay intervals the performance loss is not tolerable.

Figure 4 shows the average leakage energy savings for the STP predictor. As expected from the IPC degradation, for very small decay intervals (64 and 256 cycles), the early deactivation of entries result in no leakage energy savings at all due to the induced extra execution cycles that completely cancel whatever leakage power savings provided by the decay mechanism. However, for 1024 cycles and, particularly, for a 4K-cycle decay interval, the proposed VP decay approach obtains 52% average leakage energy savings when considering a medium size (20KB) predictor. As expected, further leakage savings can be obtained as we increase VP size but the speedup provided for bigger sizes (> 20KB) does not justify the additional resources used.

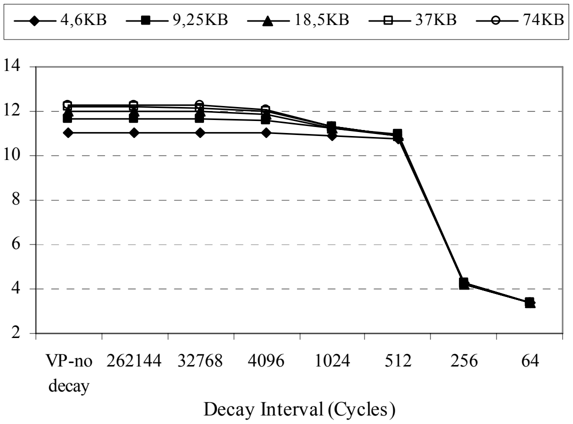

Figure 3. STP value predictor performance degradation (average SpecInt2000).

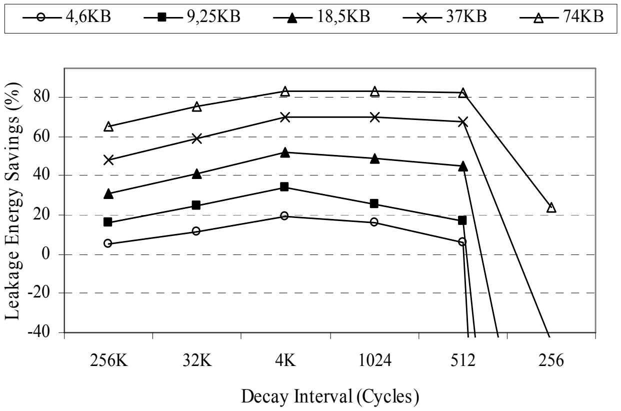

Figure 4. STP value predictor leakage energy savings (average SpecInt2000).

Figures 5 and 6 show the performance degradation and the average leakage energy savings for the FCM Value Predictor, respectively. Since the FCM is a two-level predictor (with the relevant and bigger part of the data stored in the second level table), we will be disabling both tables. We can notice a similar behavior to the STP predictor for the very small decay intervals (64 and 256 cycles), again with negative leakage energy savings due to the early deactivation of entries and the induced extra execution cycles. On the other hand, for very big decay intervals (32K and 256K-cycles), the overhead is almost zero, but we obtain very small leakage energy savings (Figure 6) since there are almost no deactivations of VP entries. However, as we reduce the decay interval length, there is an increase in leakage savings with a maximum for the 512-cycle interval. For this decay interval, a 20 KB FCM predictor obtains average leakage energy savings of 75%, showing the benefits of *Value Prediction Decay*.

Figures 7 and 8 show the performance degradation and the average leakage energy savings for the DFCM Value Predictor, respectively. We can notice that the DFCM

[2] Recall that the performance degradation is also included in the energy metrics (leakage energy = leakage power * delay).

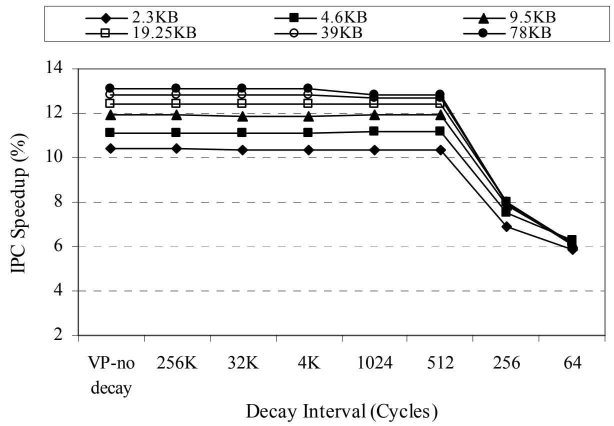

Figure 5. FCM value predictor performance degradation (average SpecInt2000).

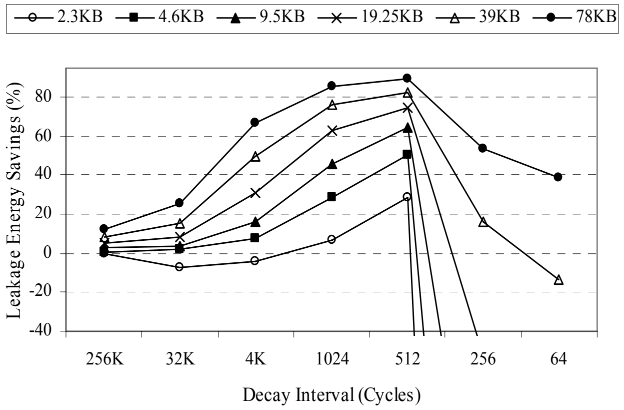

Figure 6. FCM value predictor leakage energy savings (average SpecInt2000).

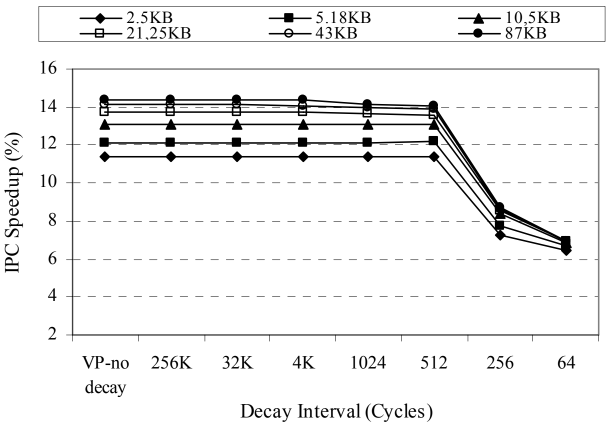

Figure 7. DFCM value predictor performance degradation (average SpecInt2000).

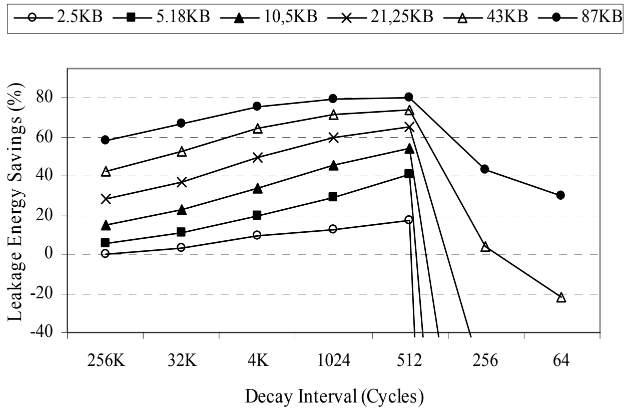

Figure 8. DFCM value predictor leakage energy savings (average SpecInt2000).

predictor behaves very similar to FCM. However, for big decay intervals, DFCM obtains better energy savings (Figure 8) due to a positive side effect when shutting DFCM entries off that results in a reduction of destructive interferences. A similar effect was also found and reported in [6] for branch predictors.

Again, as we reduce the decay interval length, there is an increase in leakage energy savings with a maximum in the 512-cycle interval. In this case, for a predictor size of about 20 KB, we obtain average leakage energy savings of 65%. For both FCM and DFCM predictors, the best energy savings are obtained for a decay interval within the 512-cycle range, unlike data caches where the best decay intervals are within the 8-Kcycle range [10], because the "average" *live time* is around 400 cycles (note also that the decay mechanism needs some additional cycles to consider that an entry has entered in a *dead state*).

Finally, Figure 9 shows the leakage energy savings breakdown for a predictor size of 20 KB and a decay interval of 1024 cycles. It can be observed that a significant amount of leakage savings are obtained when disabling VP entries during its *live time*. As commented in Section 3, there are many cases where even though an entry is *live*, the next access will be far in the future (more than 1024 cycles ahead in this experiment). In such cases, short decay intervals can obtain even further leakage savings by early disabling those entries. Figure 9 shows that, on average, half of the leakage energy savings comes from disabling entries during their *live time* and the other half comes from disabling entries during a *dead time*. Note also that the three evaluated predictors obtain a very similar leakage savings breakdown since they all are indexed in the same way (the instruction PC).

6. Conclusions

In this paper we propose *Value Prediction Decay*, a mechanism able to dramatically reduce the leakage energy of Value Predictors with negligible accuracy reduction, especially for deep-submicron high performance processor designs.

Our proposal dynamically tracks the accesses to each Value Predictor entry in order to determine if the entry has

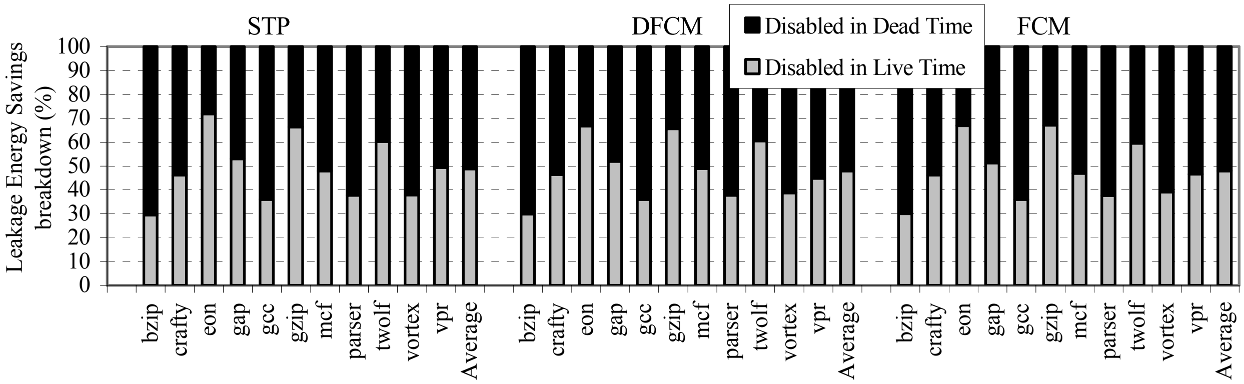

Figure 9. Leakage energy savings breakdown (20 KB predictor with decay interval of 1024 cycles).

been unused for a significant amount of time, and in that case, it switches the entry off, avoiding leakage loss. Although the VP structure is smaller than a L2 cache, hot small structures can leak more than large but cooler ones. We must fight to reduce leakage at every level, even at the smallest structures, like Value Predictors, Branch Predictors, BTBs, etc.

Experimental results have shown that both FCM and DFCM seem to be the most energy-efficient predictors achieving average leakage energy savings of 75% and 65%, respectively, for a predictor size of around 20 KB with negligible processor performance degradation when considering a decay interval of 512 cycles. We have also shown that leakage energy savings are not limited by only detecting *dead times*, since Value Predictors are structures that exhibit long periods of inactivity during an entry's *live time* which allows to early shut the entry off in order to obtain further leakage savings.

Finally, the present work tries to show how the use of low-power VP structures could still make Value Prediction a power-performance efficient mechanism suitable for modern high-end processor designs.

Acknowledgements

This work has been supported by the Ministry of Education and Science of Spain under grant TIC2003-08154-C06-03.

References

[1] J.A. Butts and G. Sohi. "A static power model for architects". In *Proc. of the 33rd Int. Symp. on Microarchitecture*, 2000.

[2] B. Calder, G. Reinman and D.M. Tullsen. "Selective Value Prediction". In *Proc. of the 26th Int. Symp. on Computer Architecture*, May 1999.

[3] K. Flautner et al. "Drowsy Caches: Simple Techniques for Reducing Leakage Power". In *Proc. of the 29th Int. Symp. on Computer Architecture*, 2002.

[4] F. Gabbay and A. Mendelson. "Speculative execution based on value prediction". *Technical Report 1080*, Technion – Israel Institute of Technology, 1997.

[5] B. Goeman, H. Vandierendonck and K. de Bosschere. "Differential FCM: Increasing Value Prediction Accuracy by Improving Table Usage Efficiency". In *Proc. of the Int. Symp. on High-Performance Comp. Architecture*, 2001.

[6] Z. Hu et al. "Applying Decay Strategies to Branch Predictors for Leakage Energy Savings". In *Proc. of the Int. Conf. on Computer Design*, Sep. 2002.

[7] Y. Li et al. "State-Preserving vs. Non-State-Preserving Leakage Control in Caches," In *Proc. of the DATE Conference*, Feb. 2004.

[8] M. Lipasti, C. Wilkerson and J. Shen. "Value locality and load value prediction". In *Proc. of the 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, Oct. 1996.

[9] P. Juang et al. "Implementing Branch-Predictor Decay Using Quasi-Static Memory Cells". *ACM Transactions on Architecture and Code Optimization*, vol. 1, June 2004.

[10] S. Kaxiras, Z. Hu and M. Martonosi. "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power". In *Proc. of the 28th Int. Symp. on Computer Architecture*, 2001.

[11] N.S. Kim, T. Austin *et al*. "Leakage Current: Moore's Law Meets Static Power". *IEEE Computer*, 2003.

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

[13] Y. Sazeides and J.E. Smith. "The predictibility of data values". In *Proc. of the 30th Annual International Symposium of Microarchitecture*, Dec. 1997.

[14] S. Yang *et al*. "An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance I-Caches". In *Proc. of the 7th Int. Symp. on High-Performance Computer Architecture*, 2001.

[15] Y. Zhang, D. Paritkh, K. Sankaranarayanan, K.Skadron and M. Stan. "HotLeakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects". *Technical Report*, Dept. of Computer Science, Univ. of Virginia, 2003.