

Applying the *zeros switch-off* technique to reduce static energy in data caches

R. Ubal, J. Sahuquillo, S. Petit and P. López

Dept. of Computing Engineering (DISCA)

Universidad Politécnica de Valencia, Valencia, Spain

raurte@inf.upv.es, {jsahuqui, spetit, plopez}@disca.upv.es

Abstract

Zeros switch-off is a leakage energy reduction technique applicable to cache memories. It works at the cache word level by removing the power supply of all or part of its most significant bytes when they store a zero, taking advantage of the high percentage of zero data bits in common programs. Experimental results, obtained by using the SPEC2000 benchmarks suite, show that the average leakage energy savings reach 60.3% with no IPC loss indeed. The proposed technique can be combined with other existing energy reduction techniques, reaching, on average, 67.3% savings with 0.5% IPC losses.

1. Introduction

Cache memories occupy a high percentage of silicon area. This fact has motivated a large number of research works to be focused on alternative organizations of conventional caches: if the cache area is reduced without losing performance, the corresponding space can be dedicated to improve other processor parts.

Recently, the situation has changed due to technology advances, and the critical problem is not the available area. On-chip level 3 caches have reached 6MB in some cases, and their area sometimes exceeds processor core size. Contrarily, current research on cache memories is mainly addressed to reduce power consumption, while trying to guarantee cycle time.

Laptops and mobile devices were the first targets of low-power devices. The limited battery energy made the low-power consumption an important requirement. While this was not the case of high-performance processors, the increasing transistor densities and clock frequencies have led to very high power consumption, where both problems of current supply and heat dissipation have to be addressed.

Hence, the problem of a high power consumption also affects, to a great extent, high performance processors, which represent an important segment of the current market. We

handle with a problem that needs effective solutions, and whose impact is greater in those processor parts which occupy an important percentage of the chip area, being the cache memory the main candidate for this aim.

We can distinguish two kinds of dissipated energy: dynamic and static energy (also called leakage energy). In CMOS circuits, the dominant consumed power is the dynamic energy, which takes place when transistors change their state. Dynamic power is proportional to the square of the supplied voltage; thus, the most common applied power reduction technique consists on reducing voltage supply. Although this is an effective technique, we still have the problem of current leakage, which is continuously consumed, even when transistors do not change their state.

Leakage energy represented a negligible problem in the past. However, time has made it acquire growing prominence, because it is directly related with the number of transistors on the chip and inversely related with the feature size. Leakage energy currently represents between 40% and 50% of the total dissipated energy [11], and is becoming higher in already upcoming 65nm fabrication technologies [1], showing a tendency to be the predominant kind of energy consumption.

This paper focuses on reducing leakage energy consumption in level 1 cache memory, although developed strategies are suitable for caches of any level. Based on the observation that most data values stored in the cache are composed of bits set to zero, we propose the *zeros switch-off* technique, that switches off the power supply to some SRAM cells.

The proposed policy applies the *gated-VDD* technique, described in [10], on contiguous sets of bytes, removing leakage consumption in the associated cache cells. *Zeros switch-off* is a non-destructive policy, since we do not lose information when switching off the chosen bytes; a special encoding system adds additional bits per cache word to indicate which cells are being powered off. *Zeros switch-off* produces high leakage energy savings without adversely impacting processor performance. An important additional feature is its compatibility with other previously developed

leakage reduction techniques.

The remainder of this paper is structured as follows. We present first other research activities related with cache power reduction in section 2. Section 3 describes our proposal and discusses its hardware implementation details. In section 4, we enumerate the steps followed to reach final energy saving results, and finally, section 5 presents some concluding remarks.

2. Related work

An important set of research work has focused on decrementing cache leakage by reducing the power supply. In this sense, two main methods have been proposed in recent literature: *gated-VDD* and *drowsy caches*.

The *gated-VDD* [10] technique proposed by Powell et al. uses a transistor to switch off the supply of the SRAM cache cells. This technique reduces the current leakage since selected lines are powered off. Of course, when lines are powered off, the stored information is lost. As a consequence, the L1 miss ratio increases, hence this technique involves additional accesses to the L2 cache, which incur additional energy dissipation.

Drowsy caches [7] are an alternative technique proposed by Flautner et al. This technique saves energy by reducing the power supply for the selected cache lines, though the supply is not removed. Proceeding in this way, the technique avoids information losses. However, these lines are placed in drowsy mode, and need to be awakened prior to accessing the data, which increases the hit latency. The advantage of this technique is that it achieves the same L1 hit ratio as conventional caches; as a hit in a drowsy line does not incur additional accesses to the L2 cache. Nevertheless, as the supply voltage is reduced, current leakage consumption is higher than the leakage introduced using the *gated-VDD* technique.

Since the switch-off policy proposed in this work is non-destructive, we decided to use the *gated-VDD* technique, which offers better leakage savings. The *gated-VDD* technique has been used previously in the *cache decay* mechanism [9] to switch off whole cache lines at regular intervals. In contrast, in this work we apply the *gated-VDD* technique to some bytes, whose choice is explained in section 3.2, when they are stored in the cache. Next, we summarize the recent research dealing with value locality, a key concept behind our switch-off policy.

2.1. Value locality

Recently, several techniques [5, 14, 12, 13, 6, 8, 3] that take advantage of the program's *value locality* have been proposed to increase performance and reduce energy consumption while maintaining (and even decreasing) transis-

tor area. Value locality is a type of locality presented by most programs that can be realized from the following observation: although current word sizes (32 or 64 bits) allow to encode an enormous range of values, only a small subset of them is used during program execution [14]. This subset has a high percentage of small, positive numbers (including zero). Due to this reason, an overwhelming majority of the bits stored in the memory elements of the processor (e.g., pipeline registers, register files, caches) are zeros [6]. In addition, it has also been observed that when there are differences among the stored values, these differences usually only affect to a few least significant bits [8].

According to these observations, the proposed techniques can be classified in two main groups. The first group deals with the high percentage of zeros [12, 3, 6]. Focusing only on zeros allows to propose simple and cost-effective techniques, which is the main reason why we take this approximation in this work.

In a recent work, Chang et al. [6] propose the *Dynamic Zero Sensitivity* scheme (DZS) that exploits the zero occurrences to reduce dynamic power consumed by cache reads. The DZS scheme prevents bitlines from discharging when reading a 0 (the common case). In contrast, our work deals with leakage consumption, which is independent of the dynamic accesses to the cache. To deal with the leakage problem, in [3], Azizi et al. propose an asymmetric cache SRAM cell where some transistors present a higher V_t , reducing leakage when the stored bit is zero. This allows to decrease the leakage consumption by a high percentage, but it can increase read access times. Although in this work we also deal with the leakage problem, our proposal is a microarchitectural solution that uses a special encoding to represent consecutive zeros, allowing us to completely switch off the affected bits and providing greater gains without excessively increasing read access times. Nevertheless, both Chang's and Azizi's proposals are orthogonal to our work, and can be therefore applied at the same time.

Another microarchitectural solution that encodes the zero values is explained in [12] to reduce dynamic power consumption (instead of leakage) by accessing just one bit instead of the whole byte when the byte is zero. As we will show in section 4.2.1, the encoding proposed in [12] is not the most appropriate one for aiming leakage reduction.

The second group of proposals [14, 13, 8] tries to detect the frequent values and to store them in special repositories (e.g., a smaller cache that stores most frequent values). The main benefit of these proposals comes from a reduction in the transistor area without affecting performance. However, the proposed designs are usually more complex. In [14], a *Frequent Value Cache* (FVC) is proposed. The results in this work show that when a small FVC is combined with a direct-mapped cache, the hit ratio obtained is similar to the one obtained by a bigger cache. In [13] a variation of the

FVC is used aiming to reduce dynamic energy consumption.

Finally, regarding the execution pipeline, González et al. [8] propose a new register bank organization based on *partial value locality*, where they do not only detect and store the most frequent values, but also find frequent partial matches of the most significant bits among values. On the other hand, R. Canal et al. [5] explore very low power pipeline designs, in which data, addresses, and instructions are compressed by maintaining only significant bytes, using two or three extension bits to indicate the significant byte positions.

3. Proposed technique: zeros switch-off

3.1. Description

The proposed policy, called *zeros switch-off*, consists on removing power supply (applying the *gated-VDD* technique) to some of the most significant bytes of a word whose value is zero. It is aimed to reduce static energy spent in data caches, without impacting processor performance, based on the fact that a large percentage of words have an important number of 0s in their most significant bits. An extra information attached to each word indicates how many bytes are switched off, so there is no information loss at all.

The reason for this value locality is that most data stored in the cache are small integer positive numbers, while an entire word size is used to store those values. This assumption has been checked by measuring, cycle by cycle, how many 0s has each word in its most significant bits. The SPEC2000 benchmark suite was used in this experiment. Figure 1 and Figure 2 show the average results, for the integer and floating-point benchmarks, respectively. We use the notation $m(x)$ to refer to the average percentage of words having 0 in their x most significant bits. For example, looking at Figure 1 one can see that $m(32)$ of the *175.vpr* benchmark is about 68%, meaning that there is a high potential to reach energy saving by applying this technique.

Notice that $m(x)$ is always greater than $m(x+1)$, since the latter is a subset of the former. For example, $m(31)$ for the *175.vpr* benchmark is about 72%. The difference between this value and $m(32)$ (i.e. 4%) is the percentage of words having exactly their 31 most significant bits set to zero, and is represented by $M(31)$. We can define $M(x)$ as follows:

$$M(x) = \begin{cases} m(32) & \text{if } x = 32, \\ m(x) - m(x+1) & \text{otherwise.} \end{cases} \quad (1)$$

Observing the results, we can expect high potential benefits in energy savings due to the *zeros switch-off* technique,

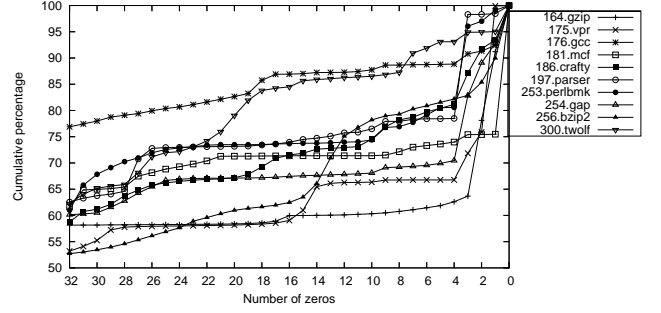


Figure 1. Distribution of average number of zeros for integer SPEC2000 benchmarks.

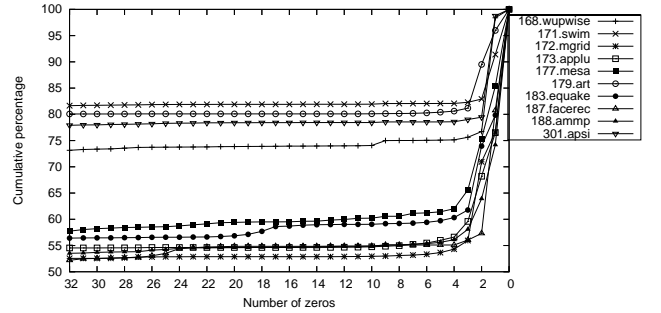


Figure 2. Distribution of average number of zeros for floating-point SPEC2000 benchmarks.

because all curves start from a high percentage, which ranges from 50% to 85%.

In order to estimate the theoretical maximum energy savings the technique could provide, let us assume an ideal algorithm able to switch off, in each cycle, all most significant bits of all words in the cache at zero hardware cost. The average number of switched off bits per cycle can be obtained by means of the following equation, where nw is the number of words in the L1 data cache. In our case, for a 64KB level 1 cache, $nw = 2K$ words, each word consisting of 32 bits.

$$(32 \cdot M(32) + 31 \cdot M(31) + \dots + 1 \cdot M(1)) \cdot nw \quad (2)$$

If we divide equation 2 by the total number of bits in the cache (nb), we obtain the percentage of switched off bits (r_{off}) of the cache in the ideal case, and therefore, the maximum possible energy saving obtained by applying this technique. For instance, the maximum energy saving for the *175.vpr* benchmark in a 64KB capacity storage cache ($nb = 64KB \cdot 8 = 512Kbits$) is given by the expression:

$$r_{off} = \frac{(32 \cdot M(32) + 31 \cdot M(31) + \dots + 1 \cdot M(1)) \cdot nw}{nb} = \frac{4.36 \cdot 10^5}{5.24 \cdot 10^5} = 83.16\% \quad (3)$$

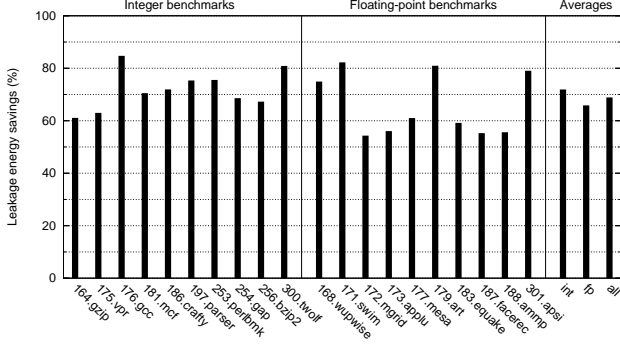


Figure 3. Theoretical maximum energy savings for SPEC2000 benchmarks

Figure 3 shows the maximum savings for some of the SPEC2000 workloads. These are theoretical values, as we are assuming that i) we are switching off all those most significant bits set to zero and ii) no additional energy overhead due to specific hardware is required to implement the technique. With ideal savings near 70%, it seems possible to reach satisfactory results, taking care of the energy due to extra control hardware.

The *zeros switch-off* technique is addressed to act over each individual word, and not over each cache line, which makes it differ of *cache decay* and *drowsy caches* techniques. On the other hand, notice that the proposed policy is orthogonal to both cited additional techniques, i.e., we can choose a design applying either *cache decay* combined with *zeros switch-off*, or *drowsy caches* with *zeros switch-off*.

3.2. Hardware implementation

In a cache memory, each bit is stored in a single SRAM cell, typically composed of six CMOS transistors [6]. Four of them constitute a latch to store the data, while the others fulfil the function of pass transistors, and permit the write and read actions over the latch. In a design that allows bits to be switched off, a new *gated-VDD* transistor is needed, which dynamically connects or disconnects the entire cell from *ground*.

Our goal is to introduce a low overhead control circuitry that switches off the cells when their contents is zero. Moreover, we must establish a low cost encoding to determine

how many cells are switched off, and thus, how many zeros must be returned in a read operation over those cells.

As a first approach, we propose to work at the byte level of a word, encoding the number of switched off bits in a 4-bit vector S ($S_3 \dots S_0$), each one representing a byte of the corresponding cache 32-bit word W . The value of S_i is connected to the *gated-VDD* transistors of the associated bit cells. Therefore, when S_i is active, the bits $W_{i*8+7 \dots i*8}$ are switched on; their power is disabled otherwise. Each bit of S is stored in a *latch*, composed of six CMOS transistors.

Figure 4 shows the hardware implementation of a) the vector S and *gated-VDD* transistors (replicated in each cache word), b) the circuit needed for a write operation (one for each write port), and c) the circuit for a read operation (one per read port).

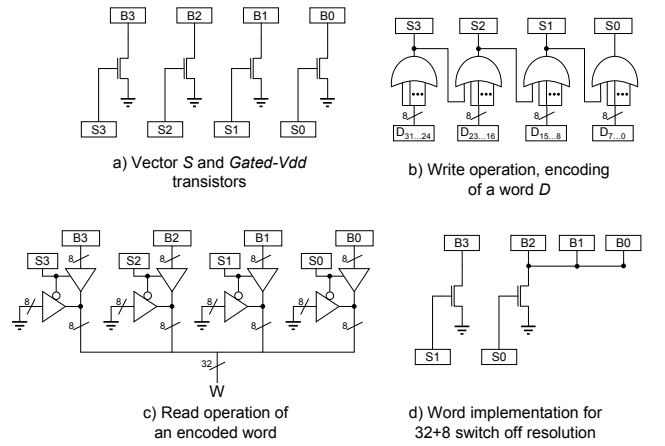


Figure 4. Hardware implementation of zeros switch-off

During the write, the elements of S are determined depending on the number of most significant bits of W set to 0. Their value responds to the logical functions $S_3 = W_{31} + W_{30} + \dots + W_{24}$, $S_2 = S_3 + W_{23} + \dots + W_{16}$, and so on. The value of S is calculated forming a chain, which increments the write latency. Nevertheless, this is not a critical issue to sustain processor performance, as the writes are not in the critical path.

During a read, the resulting word is formed according to the value of S . If S_i is 0, the corresponding bits are equal to 0, and must be taken from a *gnd* line. Otherwise, their value is the one read from the word cells $W_{i*8+7 \dots i*8}$. This function is implemented by the tristate buffers, whose activity is controlled by S . When S_i is 0, the corresponding buffer enables its *gnd* entry to be dumped into the word line. In contrast to the write circuit, whose latency is hidden to the processor, the read latency must be as short as possible as it might impact processor cycle time. Therefore, minor latency can be added to this circuit. In our design, only a

single additional gate level is added.

We can consider the possibility of decreasing the number of bits in vector S , reducing extra hardware cost accordingly. If we have a number n of bits in S which is lower than the number of bytes per word, we have to decide how to group the bytes controlled by a given S_i . This idea leads us to the concept of *switch off resolution*. Obviously, those combinations that do not switch off all bytes are not interesting.

For $n = 2$, there are three useful connections: i) S_1 connected to $W_{31...24}$ (the most significant byte) and S_0 connected to $W_{23...0}$ (the three least significant bytes); ii) S_1 connected to $W_{31...16}$ and S_0 connected to $W_{15...0}$; and iii) S_1 connected to $W_{31...8}$ (the three most significant bytes) and S_0 connected to $W_{7...0}$ (the least significant byte).

By using the (i) choice, with $S = 00$, we get the possibility of switching off the whole word (32 bits off) or, with $S = 01$, only the most significant byte (8 bits off). With $S = 11$, all bytes of the word are turned on. Notice that $S = 10$ is also possible, but it is not interesting as this combination does not correspond to a small number stored in the memory word. We will refer to resolution (i) as 32+8 (either 32 or 8 bits off, always starting from the most significant bit). Figure 4d shows the logic required at each word to implement this particular resolution.

In a similar way, choices (ii) and (iii) will be referred as 32+16 and 32+24, respectively. The benefits of a specific resolution depend on the value locality exhibited by applications and will be empirically evaluated in section 4.

For $n = 3$ (three control bits per word), there are also three connection patterns: 32+24+16, 32+24+8 and 32+16+8. Each pattern has four possible states, encoded by $S_{2...0}$: 111, 011, 001 and 000. Notice that this encoding consumes more bits than needed (four states could be encoded with just two bits) but it simplifies the implementation and is faster. Decoding vector S would impact the read access latency. Moreover, the hardware saved by using a more compact encoding would be spent, and maybe exceeded, in the decoding logic.

3.2.1. Area overhead

The incorporation of *zeros switch-off* in a cache design adds extra hardware, which occupies additional chip area. Therefore, this area must be estimated for fair comparison purposes. The area overhead is due not only to the extra bits that need to be stored, but also by power gating transistors and read/write extra hardware. Moreover, it is dependent on the switch off resolution. Table 1 represents this factor for different resolutions.

As it can be seen, the highest resolution increments the cache area by about 12%. However, experimental results shown in further sections show that the most useful resolutions will not exceed a 5.88% area overhead.

Table 1. Area overhead.

Resolutions	Bits in S	Area overhead
32	1	2.94%
32+24, 32+16, 32+8	2	5.88%
32+24+16, 32+24+8, 32+16+8	3	8.82%
32+24+16+8	4	11.76%

3.2.2. Circuit optimization

The theoretical basis underlying *zeros switch-off* leads to hardware implementations acting over combinations of most significant bits. Nevertheless, the circuits shown in figure 4 offer the possibility of switching off inner combinations of bits set to 0, without increasing hardware complexity. By releasing this restriction, the write circuit (4b) is even simplified, as chain connections among *or* gates disappear; the rest of the additional hardware remains the same.

The amount of words containing zero sequences which do not start at the MSB, that can take advantage of this new write circuit, depends on the switch off resolution, but constitutes in any case a negligible fraction (less than 1%). Thus, the proposed optimization is aimed at simplifying and accelerating the implicated encoding logic, rather than improving consumption savings.

4. Experimental framework

4.1. Simulation environment

For evaluation purposes, we implemented the proposed energy reduction technique on top of the SimpleScalar toolset [4]. On this platform, experiments were run using the SPEC2000 benchmark suite [2]. Both integer benchmarks (SpecInt) and floating-point (SpecFP) have been evaluated using the *ref* input sets. Statistics were gathered for the execution of 1 billion instructions after skipping the first million cycles. Table 2 summarizes the architectural parameters of the baseline microprocessor core.

The power model has been designed using the transistor power as consumption unit. The methodology of extracting power statistics is based in measuring in each cycle the number of transistors in the cache corresponding to switched on bits, considering data, tags and control circuitry. The relationship between the number of active transistors in a low power design and in a traditional cache design provides the percentage of energy savings, used to evaluate the effectiveness of a proposal.

Table 2. Machine Parameters.

Microprocessor core	
Issue policy	Out of order
Branch predictor type	Hybrid gShare/bimodal: Gshare has 16-bit global history plus 64K 2-bit counters. Bimodal has 2K 2-bit counters, and the choice predictor has 1K 2-bit counters
Branch predictor penalty	2 cycles
Fetch, issue, commit bandwidth	4 instructions/cycle
ROB size (entries)	64
# of Integer ALU's, multiplier/dividers	4/1
# of FP ALU's, FP multiplier/dividers	2/1
Memory Hierarchy	
L1 data cache	16KB, 4 way, 64 byte-line
L1 data cache hit latency	3 cycles
L2 data cache	512KB, 8 ways, 64byte-line
L2 data cache hit latency	18 cycles
Memory access latency	200 cycles

4.2. Experimental Results

After obtaining the zeros distribution in the data values stored in cache words, we can calculate the leakage savings by applying the zeros switch off mechanism. In this section, we first analyze which switch off resolution (i.e. number of bits required to encode the status of a word) leads to the best results. Then, we will compare the results obtained by *zeros switch-off* with *cache decay*. Finally, we will analyze the behavior of both techniques when simultaneously applied.

4.2.1. Choice of switch off resolution

Figures 5 and 6 show the net energy savings for selected switch off resolutions, taking into account the overhead of the hardware implementation discussed in section 3.2. As mentioned above, the values placed in the resolution names denote the possible number of most significant bits that may be switched off at once. Notice that 0 (none bit is switched off) is implicit in all of them. The explored resolutions range from two switch off combinations (32; only one bit in S) to five combinations (32+24+16+8; four bits in S).

In a high resolution technique, we deeply exploit the existence of different number of bits set to 0 at the beginning of the words, but we introduce, however, a high extra hardware overhead (vector S of larger size and higher number of *gated-VDD* transistors). For a low resolution technique, we decrease the switch off control circuit cost, but do not take fully advantage of the zeros switch off capabilities.

The results represented in both figures show a low resolution approach as the most appropriate option. The switch

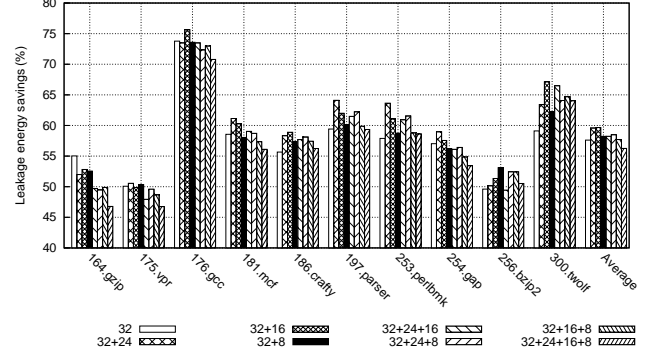


Figure 5. Leakage energy savings for integer benchmarks and different switch off resolutions

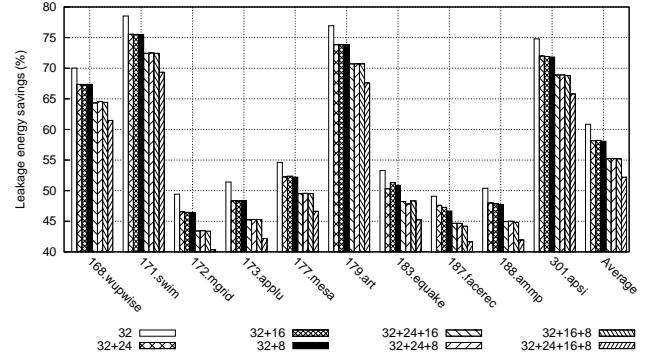


Figure 6. Leakage energy savings for floating-point benchmarks and different switch off resolutions

off resolution 32 (i.e. turn on/off all word bits) is the key to obtain good energy savings. The reason is that the implemented technique benefits mainly from entire words set to 0. Allowing to partially switch off the word (to exploit the high number of small integers) is only interesting if the associated overhead is low. For integer benchmarks, resolutions encoded with two bits (32+24, 32+16 and 32+8) improve over 32, on average. Resolutions that require more bits to encode the word state only slightly increase savings over 32 or produce even worse results.

It has already been argued that the application of *zeros switch-off* is motivated by the existence of zero or small integer numbers. However, floating-point benchmarks also experiment high benefits when applying the technique, because they make use of an important percentage of integer variables. Besides, floating-point zeros are also represented with 4 or 8 bytes set to zero, for single and double precision representation, respectively.

We can conclude that the optimal resolutions for integer

benchmarks are those which allow three switch off combinations: 32+24 (the best in 4 cases), 32+16 (the best in 2 cases) and 32+8 (the best in one case). For floating-point benchmarks, the resolution that allows two combinations (32) produces best results. A processor designer may choose one of them depending on the numerical computation field the processor is aimed for.

4.2.2. Comparison of zeros switch-off with cache decay

As mentioned above, the orthogonality of the *zeros switch-off* technique with other power reduction techniques, like *cache decay*, allows them to be combined in a single design. A new set of simulations has been performed to apply both power reduction techniques. The parameters for *cache decay* have been chosen so that we reach high energy savings with a minimal processor performance degradation. They are described next:

- *Cache decay* is a line switch off policy, based on local counters for each cache line and a global counter. The local counters range in this simulations from 3 to 0. The line is switched off when the corresponding local counter value is 0 and it receives a decrement notification. At this moment, data is lost due to the fact that *cache decay* is a non-state-preserving technique.
- Global counters range from 8191 to 0. Each time that the global counter underflows, a decrement signal is sent to all local counters. More details about *cache decay* implementation can be found at [9].

Figure 7 shows the obtained results for the three energy saving mechanisms: *cache decay*, *zeros switch-off* and a combined mechanism that applies both techniques. The *zeros switch-off* technique improves 8.2% the *cache decay* technique. If we combine both techniques, an additional 12.7% saving is obtained versus *cache decay*.

Cache decay is an aggressive leakage reduction policy, acting over whole cache lines when their life time is estimated to have ended. Nevertheless, there is a small probability of making wrong switch off decisions, causing data loss and performance degradation. As a result, IPC slightly sinks in an average factor of 0.46%. Figure 8 represents how this phenomenon affects each particular benchmark.

The main positive feature of our proposal is the improvement in energy savings without any cache latencies increment, data destruction, or induced cache misses: the information stored in the switched off bits is not lost because it remains compressed in the elements of vector S . For this reason, and taking into account that *zeros switch-off* provides, on average, higher savings than *cache decay*, a designer may be tempted to choose a raw *zeros switch-off* policy, to avoid the IPC loss incurred by state losing switch-off techniques.

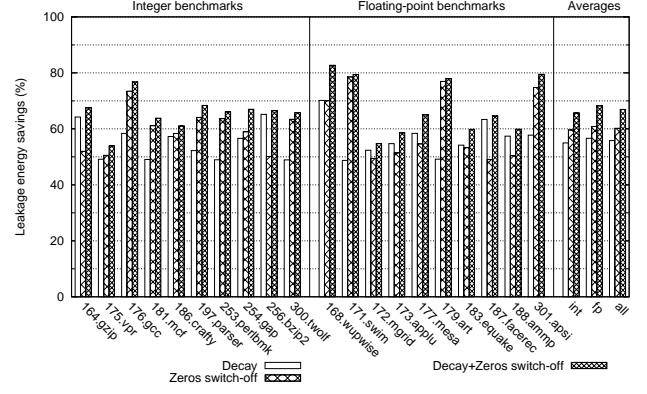


Figure 7. Leakage energy savings for cache decay, zeros switch-off, and a combination of both

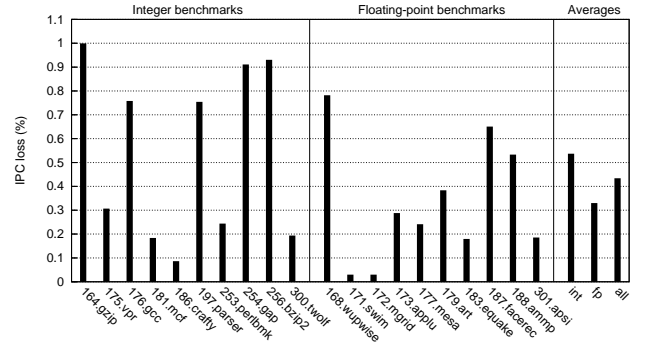


Figure 8. IPC loss when combining cache decay with zeros switch-off

5. Conclusions

In this paper, we proposed a technique (*zeros switch-off*) to reduce leakage energy in cache memories by exploiting the zero-bits distribution of program data, concentrated at the most significant parts of the words. The technique switches off all or part of the most significant bytes of a word when they store a zero, also encoding in some bits which bytes have lost power supply.

This paper concentrated on L1 data caches, where the leakage problem is more critical, but the study could be also extended to other levels of the cache hierarchy. Different mechanisms can be applied to higher level caches, like changing their transistor technology, but their stored data follow analogous distribution as in level 1, which makes L2 or L3 caches as well susceptible of implementing *zeros switch-off*.

We analyzed the potential of the proposed technique and proposed a hardware implementation. Different switch off

resolutions were explored, taking into account the overhead due to the hardware implementation. For the SPEC2000 benchmarks simulations, *zeros switch-off* provides an average energy saving of 60.3%, while a combination of *cache decay* and *zeros switch-off* reaches 67.3% savings. These results consider the energy consumed by extra control logic.

The increment of cache area depends on the selected resolution; for the most useful resolutions (e.g., 32 or 32+24), the area overhead does not exceed 5.88%, considering both extra transistors required to control voltage supply and those required to encode which bytes are switched off. With this small chip area penalty, we can conclude that *zeros switch-off* is a factible and encouraging mechanism to alleviate leakage power consumption, without affecting processor performance and transparent to the rest of processor architecture.

As for future work, we plan to extend the *zeros switch-off* technique to be applied on parallel architectures, such as multi-thread processors or CMPs (Core Multi-Processors).

Acknowledgements

This work has been partially supported by the Generalitat Valenciana under grant GV06/326 and by Spanish CICYT under Grant TIC2003-08154-C06-01.

References

- [1] Intel Pentium D 920 and Pentium D 930 Processors. http://www.intel.com/products/processor/pentium_d/.
- [2] Standard Performance Evaluation Corporation. <http://www.spec.org/cpu2000/>.
- [3] N. Azizi, A. Moshovos, and F. N. Najm. Low-leakage asymmetric-cell sram. *IEEE Transactions on Very Large Scale Integration Systems*, 11(4), August 2004.
- [4] D. Burger and T. Austin. The simplescalar tool set, version 2.0. *Computer Architecture News*, 25(3), 1997.
- [5] R. Canal, A. Gonzalez, and J. Smith. Very low power pipelines using significance compression. *IEEE/ACM International Symposium on Microarchitecture (MICRO-33)*, December 2000.
- [6] Y. J. Chang and F. Lai. Dynamic zero-sensitivity scheme for low-power cache memories. *IEEE Micro*, 25(4):20–32, July 2005.
- [7] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: Simple techniques for reducing leakage power. *Proceedings of the 29th International Symposium on Computer Architecture*, May 2002.
- [8] R. González, A. Cristal, D. Ortega, A. Veidenbaum, and M. Valero. A content aware integer register file organization. *Proceedings of the 31st International Symposium on Computer Architecture*, June 2004.
- [9] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: Exploiting generational behavior to reduce cache leakage power. *Proceedings of the 28th Annual International Symposium on Computer Architecture*, July 2001.
- [10] M. Powell, S. H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar. Gated-vdd: A circuit technique to reduce leakage in deep-submicron cache memories. *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, July 2000.
- [11] D. Rittman. Power optimization within nanometer designs. *System Design Frontier, Shanghai Hometown Microsystems Inc.*, May 2005.
- [12] L. Villa, M. Zhang, and K. Asanović. Dynamic zero compression for cache energy reduction. *Proceedings of the 33rd International Symposium on Microarchitecture*, December 2000.
- [13] J. Yang and R. Gupta. Energy efficient frequent value data cache design. *Proceedings of the 35th International Symposium on Microarchitecture*, November 2002.
- [14] Y. Zhang, J. Yang, and R. Gupta. Frequent value locality and value-centric data cache design. *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, November 2000.