Leakage Current Reduction in Data Caches on Embedded Systems

R. Ubal, J. Sahuquillo, S. Petit, H. Hassan, and P. López
Department of Computer Engineering (DISCA)
Universidad Politécnica de Valencia, Spain
raurte@gap.upv.es

Abstract

Nowadays, embedded systems can be found in a wide range of pervasive devices (e.g., smart phones, PDAs, or video/digital cameras). These devices contain large cache memories, whose power consumption can reach about 50% of the total spent energy, from which leakage energy is the predominant fraction in current technologies. This paper proposes a technique to reduce leakage energy consumption in data caches on embedded systems, which is based on the fact that most stored bits take a logical value of zero. The proposed technique has been evaluated on a model of a contemporary high-end embedded microprocessor, namely the ARM Cortex A8 processor, executing a set of standard embedded benchmarks. Experimental results show that leakage energy savings reach about 40% with no IPC loss.

1 Introduction

Pervasive embedded computers incorporate large memories that consume high amounts of energy. Moreover, the battery capacity in such devices is strictly limited and it is preferable to reduce the energy consumption of the components rather than increasing the battery capacity, because it results in a reduction of the dimensions, weight, and cost of the device. Therefore, energy consumption has become a major concern when designing embedded computing systems. As a consequence, research efforts on energy optimization techniques for the design of mobile embedded devices are required [1].

Several studies have shown that cache memories account for about 50% of the total energy consumed in embedded systems [2]. Furthermore, embedded memories occupy, on average, more than 50% of the active area in SoCs (Systems on a Chip) [3]. Therefore, power consumption, both dynamic and static (also called leakage energy), is a key concern in cache memories [4].

In traditional CMOS circuits, the dominant consumed power was the dynamic energy, which takes place when transistors change their state. Dynamic power is proportional to the square of the voltage supply; consequently, the most commonly applied power reduction technique consists of reducing voltage supply. Although this is an effective technique, it does not solve the problem of leakage current, which is continuously consumed, even when transistors do not change their state. Leakage energy represented a negligible problem in the past. However, newer technologies have 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 [5], and is becoming higher in already upcoming 65 nm fabrication technologies [6] [7], showing a tendency to be the predominant kind of energy consumption.

This paper deals with reducing leakage energy consumption in first-level cache memories, although developed strategies are suitable for caches at any level. Based on the observation that most data values (mainly integer values) stored in the cache are composed of bits set to zero, we propose the *zeros switch-off* (ZSO) technique, that switches off the power supply to some *chunks* of SRAM cells, using the gated-VDD technique [8]. ZSO is a non-destructive policy, so it produces high leakage energy savings without adversely impacting processor performance.

Although the proposed technique can be applied to any kind of microprocessor, this paper focuses on embedded systems. To this end, experiments evaluate a standard set of embedded workloads, with a pipeline resembling a contemporary manufactured embedded microprocessor (i.e., ARM Cortex A8 [7]), and different cache sizes (which is a major concern in embedded systems). Experimental results show that leakage energy savings reach about 40% with no IPC loss indeed.

2 Related Work

The reduction of leakage energy in data caches has become a key focus of interest for embedded systems since leakage power has emerged as a critical design issue. Two main methods have been proposed in recent literature: gated-VDD and drowsy caches, depending on whether they destroy the cell contents or not.

The gated-Vdd [8] technique proposed by Powell et al. uses a pass transistor to cut the power supply of the SRAM cache cells, so the involved leakage current is practically removed. Of course, when lines are powered off, the stored information is lost if there is no previous encoding of the data. As a consequence, the L1 miss ratio increases, involving additional accesses to the L2 cache, which incur, in turn, additional energy dissipation.

Drowsy caches [9] are an alternative technique proposed by Flautner et al. This technique saves energy by reducing the power supply for the selected cache lines, without removing it completely. Drowsy caches reach lower savings, but information loss is avoided. However, these lines are placed in drowsy mode, and need to be awakened prior to accessing the data, which increases the hit latency.

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 [10] to switch off whole cache lines at regular intervals. In [11], a method is proposed to dynamically determine the time interval length for deactivating cache lines. In contrast, in this work we apply the gated-Vdd technique to some bytes, whose choice is explained in Section 4, when they are stored in the cache.

3 Value Locality

Several techniques [12] [13] [14] [15] [16] [17] [18] dealing with value locality, a key concept behind our switch-off policy, have been proposed. These techniques take advantage of value locality present in programs to increase performance and reduce energy consumption, while maintaining, or even decreasing, the design 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 [13]. This subset has a high percentage of small, positive numbers, including zero. For 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 [16]. In addition, it has also been observed that differences among the stored values usually only affect to a few least significant bits [17].

According to these observations, the proposed techniques can be classified in two main groups. The first group deals with the high percentage of zeros [14] [18] [16]. Chang et al. [16] proposed 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). To deal with the leakage problem, Azizi et al. [18] 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. Both Changs and Azizis proposals are orthogonal to the proposed technique, so they can be applied jointly in the same design.

The second group of proposals [13] [15] [17] [4] tries to detect the frequent values and to store them in special repositories. 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 [13], a Frequent Value Cache (FVC) is proposed, which can be combined with a direct-mapped cache, reaching a similar hit ratio as the one obtained with a bigger cache. In [15] a variation of the FVC is used aiming to reduce dynamic energy consumption. Patel et al [4] present a technique that exploit the fact that during the program lifetime a few data values tend to exhibit both spatial and temporal locality in cache. Those lines whose lifetime is highly overlapped are turned off and placed in a small low-leakage buffer located next to the data cache. Their proposal is evaluated in a like ARM 7 embedded processor showing leakage savings of about 18.35%.

4 Proposed Technique: *zeros switch-off*

4.1 Description

The baseline proposed technique, called *Zeros Switch-Off* (ZSO), is aimed at reducing static energy consumed in data caches, by removing power supply to some SRAM cells storing a logical value of zero. Extra bits are attached to each word, indicating which of its bits are switched off, so no information loss occurs at all. The obtained benefits are motivated by the value locality, i.e., a high percentage of the stored bits are zero, and they are mostly concentrated in the most significant part of the words.

The main reason for the occurrence of sequences of most significant bits set to zero is that most words stored in the cache are small integer positive numbers or integer/floating-point zeros. As a first experiment, we explored the value locality of the standard set of embedded benchmarks, by measuring cycle by cycle the length of the zeros sequences in the stored words, for a 16KB first-level data cache. Figure 1 depicts this distribution, and it is to be interpreted as the average fraction of words (Y-axis) with a specific number of most significant bits set to 0 (X-axis).

Notice that all curves in Figure 1 have a positive slope, since all words containing a sequence of x most significant bits set to 0 do also contain such a sequence of length $x-1$. For instance, the curve corresponding to the *jpeg_enc* benchmark shows that there is an average of about 61% of the stored words with all their 32 bits set to zero. As one can see, all curves evidence a high or reasonable value locality that can be exploited with ZSO.

4.2 Hardware Implementation

In cache memories, each bit is stored in a single SRAM cell, typically composed of six CMOS transistors [16]. Four of them constitute a latch to store the data value, while the others fulfil the function of pass transistors, which 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.

The proposed implementation enables to switch off sequences of bits set to zero, working at the granularity of entire bytes. For this purpose, a vector S of 4 bits ($S_3 ... S_0$)

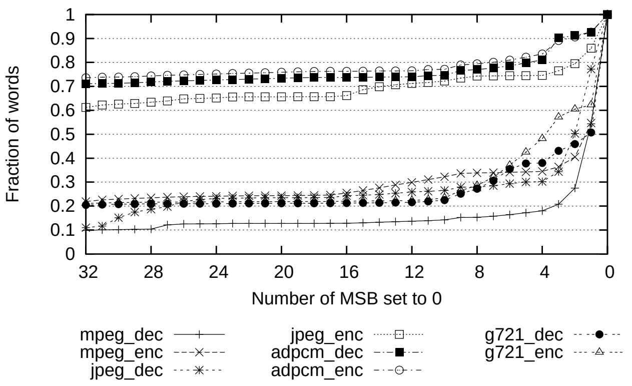
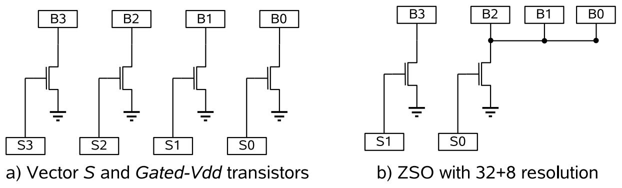


Figure 1. Distribution of zeros in the most significant part of the words.



B3
B2
B1
B0
S3
S2
S1
S0
a) Vector S and Gated-Vdd transistors
b) ZSO with 32+8 resolution

Figure 2. Word representation for different switch off resolutions.

is associated to each stored word, where each bit S_i controls the power supply of the byte i in the corresponding word. Each signal S_i serves as input for a pass transistor that feeds a specific byte of the stored word, as shown in Figure 2a. For example, an assignment of value 0011 to vector S would cause bytes 3 and 2 of the stored word to be switched off, while bytes 1 and 0 would keep power supply.

Switch off resolution. The flexibility of this implementation allows to vary the combination of stored bytes affected by a specific bit S_i. A particular combination is known as *switch off resolution*, and will be represented as X_1+X_2+...+X_n, being n the number of bits in vector S. The occurrence of X_i in the switch off resolution means that the associated hardware implementation (combination of bits S_i with stored bytes) allows to switch off a sequence of zeros comprising the X_i most significant bits. For example, resolution 32+8 allows to switch off either all 32 bits of any stored word, or only its 8 most significant bits.

The hardware implementation of resolution 32+8 is shown in figure 2b. In this case, vector S is composed of two bits. A value of $S = 00$ causes the whole word to be switched off (both gated-VDD transistors are cut); a value of $S = 01$ switches off only the 8 most significant bits of the associated word (only the left transistor is cut), while a value of $S = 11$ enables the entire word to keep power supply. Notice that higher resolutions (those with more components in S) enable zeros sequences to be captured with finer granularity, but introduce a higher overhead both in area and power. Further experiments focus on exploring different resolutions to find a trade-off solution.

ZSO Read/write circuits. A modification in the hardware representation of the data requires an adaptation of the read and write circuits. During a write operation of word W (Figure 3a), the elements of vector S must be updated. They are determined depending on how many bits of W are set to zero in the most significant side. For a resolution of 32+24+16+8 (initial approach shown in Figure 2a) each S_i must be computed using the logical functions $S_3 = W_{31} + W_{30} + ... + W_{24}$, $S_2 = W_{31} + ... + W_{16}$, $S_1 = W_{31} + ... + W_8$ and $S_0 = W_{31} + ... + W_0$. In other words, an element S_i must take a value of zero only if all its associated bits in the stored word are zero, and all elements S_j are also zero, for $j > i$.

During a read of word W (Figure 3b), the resulting word must be composed according to the value of S. If S_i is 0, the associated bits are switched off, so they are taking a value of 0, which can be read from a *gnd* line. Otherwise, their value is read from the actual SRAM cells. 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 adversely impact the processor cycle. Therefore, minor latency can be added to this circuit. In our design, only a single additional gate level is added.

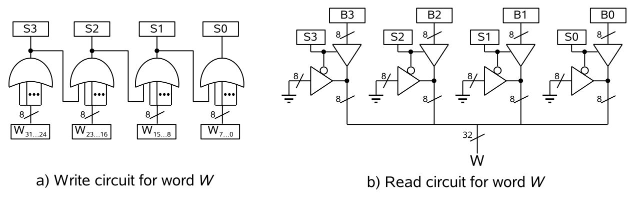


Figure 3. Hardware implementation of read/write circuits.

Full-ZSO. The discussed baseline ZSO approach tries to take profit of the fact that most data stored in the cache are composed of small integer numbers or zeros, so it acts over sequences of most significant bits. Nevertheless, the hardware representation of a word (Figure 2) as well as the implementation of the read/write circuits (Figure 3) can be adapted to switch off any sequence of bits set to zero, regardless they are continuous or not, so vector S can take any value between 00...0 and 11...1.

This enhancement constitutes a more flexible and effective technique, referred to as full-ZSO. Notice that the fact of accepting inner (non contiguous) sequences of zeros even simplifies the proposed circuitry for the write operation, because the OR-gate chain is removed (see Figure 4). The rest of the ZSO support hardware (word representation and read circuit) remains unaltered for full-ZSO.

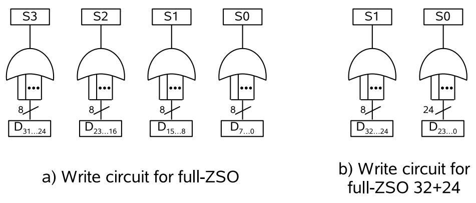

S3
S2
S1
S0
8
8
8
8
D31...24
D23...16
D15...8
D7...0
a) Write circuit for full-ZSO
S1
S0
8
24
D32...24
D23...0
b) Write circuit for full-ZSO 32+24

Figure 4. Write circuit for full-ZSO

Table 1. Machine Parameters.

Microprocessor core	
Issue policy	In order
Branch predictor	Two-level global history 256 entries BTB, 4096 2-bit saturating counters GHB 13 cycles misprediction
Issue bandwidth	2 instructions/cycle
# of Integer ALU's, multiplier/dividers	1/1
# of FP ALU's, FP multiplier/dividers	1/1
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

5 Experimental Results

5.1 Simulation Environment

The proposed techniques ZSO and full-ZSO have been evaluated on top of the simulation framework Multi2Sim [19], using benchmarks *mpeg*, *jpeg*, *g721* and *adpcm*, and running the full encoding and decoding module for each of them. Table 1 summarizes the architectural parameters of the baseline microprocessor core, that have been chosen resembling the embedded ARM microprocessor.

The leakage power model has been designed using the transistor leakage power as consumption unit. In each cycle, we measure the number of transistor corresponding to switched on bits, considering cache tags, data and control circuitry, including gated-Vdd transistors. The ratio 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 our proposals.

5.2 ZSO: Switch off Resolution

This experiment is aimed at exploring the net energy savings provided by ZSO with different switch off resolutions, taking into account the overhead of the hardware implementation discussed in Section 4.2. Associated results are shown in Figure 5. 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).

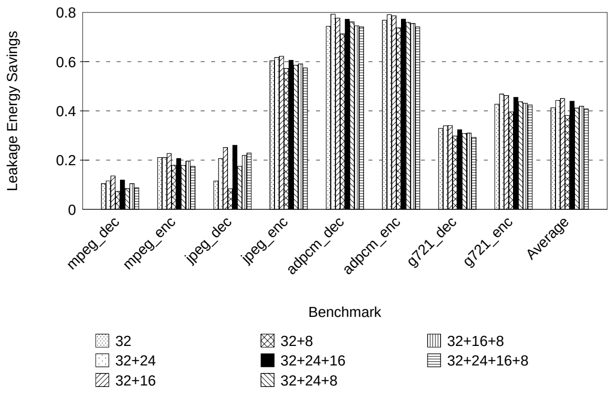


Figure 5. Leakage energy savings for different ZSO resolutions.

In a high resolution technique, we deeply exploit the existence of different number of bits set to 0 at the most significant part 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.

Our results show low resolution approaches as appropriate options. The switch 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 mainly benefits 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. Observing the block of bars containing the average values, we can conclude that the best resolutions for the selected benchmarks are 32+24, 32+16 and 32+24+16.

5.3 ZSO versus full-ZSO

As explained in Section 4.2, full-ZSO enhances the basic ZSO in the sense that not only sequences of most significant bits set to zero can be switched off, but also inner or least significant sequences of bits can lose power supply. This fact has conceptually no relationship with the initial motivation of ZSO, i.e. the distribution of values in the data cache, but it effectively takes advantage of the sporadic existence of words containing such sequences.

This experiment is aimed at measuring the benefits of full-ZSO compared with the baseline ZSO. Figure 6 shows the leakage energy savings for different switch off resolutions using both techniques, and extracting directly the average values for all presented benchmarks. As one can appreciate, there is no drastic difference between them. On one hand, resolution 32 imposes no difference at all, since only the entire words can be switched off. On the other hand, results show that higher resolutions (those with more switch off combinations) obtain more benefits when implementing full-ZSO.

The most important observation is the contrasting behaviour of the resolution increase in the case of ZSO and

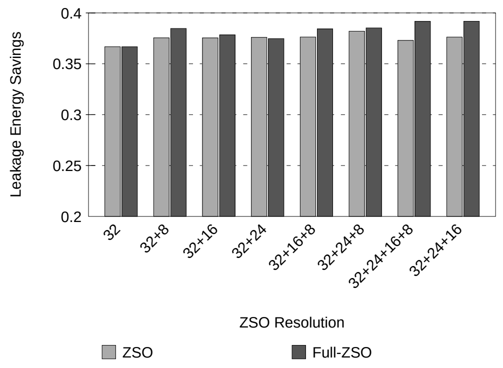


Figure 6. Comparison of ZSO vs full-ZSO.

full-ZSO. In the former, an increase of the switch off resolution further than $n = 3$ makes the overhead dominant and causes leakage energy savings to wane, as shown in 5.2. In the latter, higher resolutions enable to capture shorter intermediate sequences of bits set to zero, and maintain higher leakage energy savings even for $n >= 3$. For further experiments, we choose full-ZSO 32+24+16 as the baseline technique.

5.4 Cache decay vs full-ZSO

Cache decay is a leakage energy reduction technique based on entire cache blocks, while full-ZSO acts over single words. As mentioned above, both techniques can be combined, forming a low-power design that deactivates power supply to whole blocks when their lifetime is estimated to have expired, and to word fragments whose bits conform a sequence of zeros. In this section, we compare the *cache decay* with full-ZSO 32+24+16 technique and a combination of both.

The parameters of *cache decay* have been chosen as suggested in [10]. There is a global counter for the whole cache, which ranges from 8191 to 0. Each time the global counter underflows, a decrement signal is sent to all local counters, one per cache block, which range from 3 to 0. When a local counter underflows, the associated cache block is switched off. On the other hand, a local counter is reset each time the corresponding cache block is accessed.

On average, full-ZSO reaches almost 40% leakage energy savings, *cache decay* achieves about 70% and the combination of both obtains nearly 80%. Notice that *cache decay* provides higher savings than full-ZSO in standard embedded benchmarks. This behaviour is motivated by the high spatial and temporal locality of these programs, which can be exploited by *cache decay* to switch off cache zones that are not accessed in long time. However, the estimation of blocks lifetime made by *cache decay* can occasionally be wrong, causing decayed blocks to be accessed again, and incurring extra cache misses and accesses to other cache memories or main memory. The IPC factor is approximately decreased in 2% for our benchmarks when applying *cache decay* due to wrong switch off decisions.

A designer may select a raw full-ZSO implementation when it is critical to maintain processor performance, while a raw *cache decay* technique would be appropriate if an IPC loss is not critical and is compensated by the extra leakage energy savings. Additionally, a joint full-ZSO/*cache decay* design may be suitable for a very low power design if there are not strong chip area restrictions, and extra hardware for both techniques can be feasibly implemented.

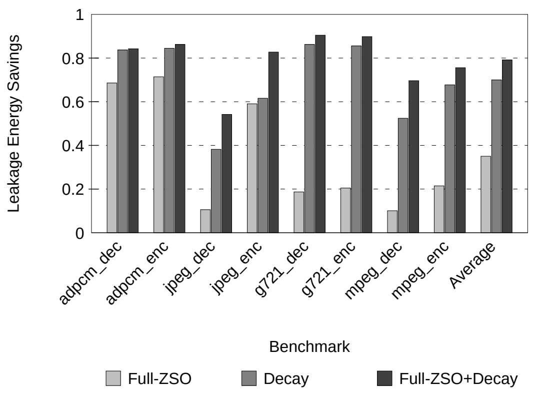


Figure 7. Leakage energy savings for full-ZSO and cache decay.

5.5 Cache size

Cache memories consume an important fraction of the total chip energy, and this fraction is higher for larger cache sizes. The size of the level 1 data cache also affects considerably the obtained leakage energy savings for both techniques. In this experiment, we studied the behaviour of proposed policies when the cache size ranges from 1KB to 64KB. As Figure 8 shows, full-ZSO obtains higher savings than *cache decay* when the cache size is so small, that continuous accesses prevent most *cache decay* counters from underflowing. From 4KB on, *cache decay* saves more leakage energy, also incurring its characteristic IPC loss. For all cache sizes up to 64KB, the combination of full-ZSO with *cache decay* constitutes the lowest power design, combining the advantages of both techniques (switch off at block and word level) jointly with their disadvantages (spend of chip area and IPC loss).

6 Conclusions

Energy consumption is a major concern when designing current microprocessors, and takes higher importance in the field of pervasive embedded systems due to battery constraints. On the other hand, cache memories conform an important percentage of the total die area, as well as a considerable fraction of the total power consumption. The total power spent in a system can be classified in dynamic and static (or leakage) energy, whose amount is similar in current technologies. In this paper, we proposed *zeros switch-off* (ZSO) as a technique to reduce leakage energy in on-chip caches for embedded processors.

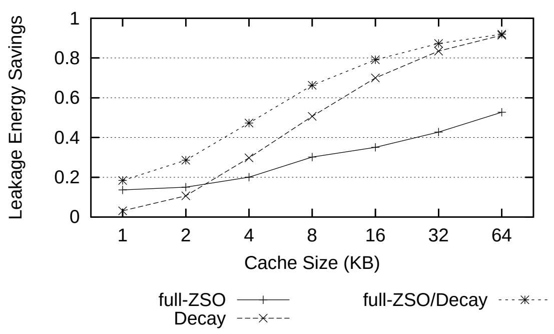


Figure 8. Savings for full-ZSO and cache decay for different cache sizes.

The proposed technique is orthogonal to most existing proposals aimed at reducing leakage or dynamic energy. This feature has been evaluated by combining our proposal with the *cache decay* technique. Experimental results show that ZSO reaches average leakage energy savings of about 40% for the standard set of embedded benchmarks, while it contributes in about a 10% in increasing them when combined with *cache decay*.

Acknowledgements

This work was supported by Generalitat Valenciana under Grant GV06/326, Spanish CICYT under Grant TIN2006-15516-C04-01 and by CONSOLIDER-INGENIO 2010 under Grant CSD2006-00046.

References

[1] M. Verma, L. Wehmeyer, and P. Marwedel. Cache-Aware Scratchpad-Allocation Algorithms for Energy-Constrained Embedded Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10), October 2006.

[2] C. Zhang, F. Vahid, and W. Najjar. A Highly Configurable Cache for Low Energy Embedded Systems. *ACM Transactions on Embedded Computing Systems*, 4(2), May 2005.

[3] Semiconductor Industry Association. The International Technology Roadmap for Semiconductors (ITRS), 2003. *http://public.itrs.net/Files/2003ITRS/Home2003.htm*.

[4] K. Patel, L. Benini, E. Macii, and M. Poncino. STV-Cache: A Leakage Energy-Efficient Architecture for Data Caches. *17th Edition of ACM Great Lakes Symposium on VLSI*, 2006.

[5] D. Rittman. Power optimization within nanometer designs. *System Design Frontier, Shanghai Hometown Microsystems Inc.*, May 2005.

[6] *Intel Pentium D 920 and Pentium D 930 Processors. http://www.intel.com/products/processor/pentium_d/.*

[7] ARM limited. White Paper: Architecture and Implementation of the ARM Cortex-A8 Processor. *http://www.arm.com/products/CPUs/ARM_Cortex-A8.html*.

[8] 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.

[9] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: Simple techniques for reducing leakage power. *Proceedings of the* 29^{th} *International Symposium on Computer Architecture*, May 2002.

[10] 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.

[11] H. Zhou and M.C. Toburen and E. Rotenberg and T. M. Conte. Adaptive Mode Control: A Static-Power-Efficient Cache Design. *ACM Transactions on Embedded Computing Systems*, 2(3), August 2003.

[12] R. Canal, A. Gonzalez, and J.E. Smith. Very Low Power Pipelines Using Significance Compression. *IEEE/ACM International Symposium on Microarchitecture (MICRO-33)*, December 2000.

[13] C. Zhang and J. Yang and F. Vahid. Low Static-Power Frequent-Value Data Caches. *DATE04: Design, Automation, and Test in Europe*, February 2004.

[14] L. Villa, M. Zhang, and Krste Asanović. Dynamic Zero Compression for Cache Energy Reduction. *Proceedings of the* 33^{rd} *International Symposium on Microarchitecture*, December 2000.

[15] J. Yang and R. Gupta. Energy Efficient Frequent Value Data Cache Design. *Proceedings of the* 35^{th} *International Symposium on Microarchitecture*, November 2002.

[16] Y. J. Chang and F. Lai. Dynamic Zero-Sensitivity Scheme for Low-Power Cache Memories. *IEEE Micro*, 25(4):20–32, July 2005.

[17] R. González, A. Cristal, D. Ortega, A. Veidenbaum, and M. Valero. A Content Aware Integer Register File Organization. *Proceedings of the* 31^{st} *International Symposium on Computer Architecture*, June 2004.

[18] 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.

[19] www.gap.upv.es/˜raurte/multi2sim.html. R. Ubal Homepage – Tools – Multi2Sim.