

Contents lists available at [SciVerse ScienceDirect](http://SciVerse.Sciencedirect.com)

## INTEGRATION, the VLSI journal

journal homepage: [www.elsevier.com/locate/vlsi](http://www.elsevier.com/locate/vlsi)

## Circuit design of a dual-versioning L1 data cache

Azam Seyedi<sup>a,b,\*</sup>, Adrià Armejach<sup>a,b</sup>, Adrián Cristal<sup>a,c</sup>, Osman S. Unsal<sup>a</sup>, Ibrahim Hur<sup>a</sup>, Mateo Valero<sup>b</sup><sup>a</sup> Barcelona Supercomputing Center, Spain<sup>b</sup> Universitat Politècnica de Catalunya, Spain<sup>c</sup> IIIA-Artificial Intelligence Research Institute CSIC, Spanish National Research Council, Spain

## ARTICLE INFO

## Keywords:

Data cache design  
Dual-versioning  
Optimistic concurrency  
Parallelism

## ABSTRACT

This paper proposes a novel L1 data cache design with dual-versioning SRAM cells (dvSRAM) for chip multi-processors that implement optimistic concurrency proposals. In this cache architecture, each dvSRAM cell has two cells, a main cell and a secondary cell, which keep two versions of the same logical data. These values can be accessed, modified, moved back and forth between the main and secondary cells within the access time of the cache. We design and simulate a 32 KB dual-versioning L1 data cache and introduce three well-known use cases that make use of optimistic concurrency execution that can benefit from our proposed design.

© 2011 Elsevier B.V. All rights reserved.

## 1. Introduction

Tremendous progress in architecture has made chip multi-processors increasingly common. To benefit from the additional performance offered by these multi-core chips, various novel architecture implementations have been proposed, such as speculative multithreading [1], lock elision [2], or hardware transactional memory [3]. All of these proposals leverage optimistic concurrency, by assuming that conflicting data accesses will not occur; in case a conflict occurs, all tentative data updates have to be undone. Thus, multiple versions of the same data have to be maintained by the L1 data cache.

The circuit design of a cache, called dual-versioning L1 data cache (dvSRAM), which supports all these different proposals, has been proposed recently [4,5]. The main characteristic of the dvSRAM cache is the inclusion of two cells in each dvSRAM cell, main cell and secondary cell, which keep two versions of the same data. These two values can be accessed separately or synchronously, modified and exchanged within the cache access time using defined operations supported by the cache. Exchange circuits are introduced to connect main cell and secondary cell to each other. When the exchange circuits are inactive, the main cell and secondary cell are isolated from each other.

In Section 2 we explain a detailed circuit design of 32 KB dual-versioning SRAM (dvSRAM) L1 data cache, a four-way set associative cache, with 64B data lines, two clock cycles access time and 45 nm Predictive Technology Model [6] at 2 GHz processor frequency. We propose a complete description of the internal structure of each array and details of all its components, such as address decoders, control signal generator units, data and word-line buffers, sub-array cells, and necessary peripheral circuitries. We simulate both the dvSRAM and the typical SRAM array with HSPICE 2003.03, design the layouts [7], and calculate dynamic and static energy consumptions and access times for all the operations.

In Section 3 we discuss how the aforementioned optimistic concurrency based systems can benefit from the efficient dual-versioning provided by the dvSRAM. Moreover, we evaluate one of the systems using state-of-the-art baseline and benchmarking suite and show that significant speedups are achieved with acceptable overall energy consumption. In Section 4 we discuss the related works, and finally we conclude in Section 6.

## 2. dvSRAM design details

In this section, we start with dvSRAM cell design. We describe the cell circuit, and we introduce the available operations of the dvSRAM array structure. Then we describe the different blocks that form the dvSRAM array structure in detail.

## 2.1. Dual-versioning cell design

Fig. 1 depicts the structure of our proposed dvSRAM cell, which is composed of two typical 6 T SRAM cells [8]: the main

\* Corresponding autor at: Barcelona Supercomputing Center, Building Torre Girona, Jordi Girona, 31, 08034 Barcelona, Spain. Tel.: +34 93 4137716; fax: +34 93 413 77 21.

E-mail addresses: azam.seyedi@bsc.es (A. Seyedi), adria.armejach@bsc.es (A. Armejach), adrian.cristal@bsc.es (A. Cristal), osman.unsal@bsc.es (O.S. Unsal), ibrahim.hur@bsc.es (I. Hur), mateo.valero@bsc.es (M. Valero).

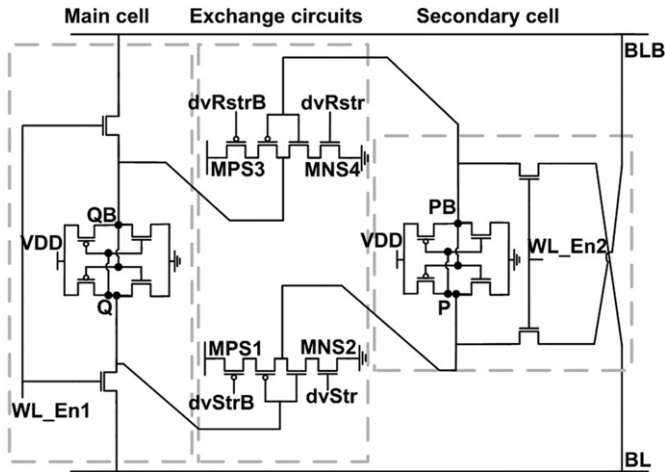


Fig. 1. Circuit schematic of the proposed dvSRAM cell: a main cell with a secondary cell and exchange circuits.

Table 1  
Brief description of the dvSRAM cell operations.

Operation	Description
Read	Reading from a main cell by activating WL_En1
Write	Writing to a main cell by activating WL_En1
dvStr	$\sim Q \rightarrow P$ : Storing the main cell to the secondary cell by activating dvStr
dvRstr	$\sim PB \rightarrow QB$ : Restoring the secondary cell to the main cell by activating dvRstr
dvWrite	Writing to the secondary cell by activating WL_En2
dvRead	Reading from the secondary cell by activating WL_En2
dvBWrite	Writing to both cells simultaneously by activating WL_En1 and WL_En2
dvStrAll	Storing all main cells to their secondary cells simultaneously

cell and the secondary cell. These two cells are connected via two tri-state inverters [8]; we call them exchange circuits, which are two inverters coupled with two pairs of nMOS and pMOS transistors. Each of main and secondary cells keeps different versions of same data. The only time they are not isolated is when the exchange circuits are active, and the data of the main cell stores to the secondary cell, or the data of the secondary cell restores to the main cell. When the signals dvRstr and dvStr are low and the exchange circuits are not active, there is more than one 'off' transistor in the nMOS or pMOS stacks in the exchange circuits, which leads to a significant reduction in the subthreshold leakage current that flows through them [9,10]. This effect is known as the stacking effect [11].

In Table 1, we briefly explain dvSRAM cell operations. Read and Write act like read and write operations in a typical SRAM cell. Other operations are created based on our goal in this paper. When dvStr is high, transistors MPS1 and MNS2 are turned on, and the exchange circuit acts as an inverter by inverting Q to P. The secondary cell, which is formed by two back to back inverters keeps the value of P when dvStr is low, and it inverts P to PB, so that PB has the same value as Q. Similarly, when dvRstr is high, PB in the secondary cell is inverted to QB and converted to Q, so that previously saved data in the secondary cell can be recovered. Note that dvStr operation acts just for a line; when we need to store all the cells to their secondary cells simultaneously, we use dvStrAll operation. We use dvRead to read the data from the secondary cell and dvWrite to write to the secondary cell. Finally, we use dvBWrite to write to both main and secondary cells

simultaneously. In Section 2.2.2, we describe how these signals are generated.

Main and secondary cells have similar sizes for each dvSRAM cell; BL is connected to Q and PB and BLB is connected to QB and P via pass gate transistors for symmetric behavior. Transistor sizing is an important point in dvSRAM cell design because two SRAM cells retain two different data values. An update to one of the cells should not lead to any changes in the preserved data value of the other cell, so the leakage current of the exchange circuits should be as low as possible in order to avoid data values to change due to leakage current.

## 2.2. dvSRAM array structure design

In this subsection we describe the high-level organization of the dvSRAM array, considering each part of its structure.

### 2.2.1. Brief description of the whole array structure

An appropriate cache modeling tool that chooses the optimal number of sub-banks, size and orientation is Cacti [12]. We use Cacti to determine the optimal number and size of dvSRAM array components and find the cache architecture with optimal access time and power consumption. For the L1 data cache configuration we assume 32 KB, four way, 64 byte lines, two clock cycles access time; and we use the 45 nm Predictive Technology Model [6]. For a one bank array, Cacti suggests two identical sub-banks, one mat for each sub-bank and four identical sub-arrays in each mat as can be seen in Fig. 2 [13]. The number of sub-banks and mats in each sub-bank are the most important results we exploit from the Cacti suggestions ( $N_{sub-banks} = N_{dbl}/2$ ,  $N_{mats-in-sub-banks} = N_{dwl}/2$  [12]). The address decoder and control signal generator units are placed in the middle part of the array and necessary drivers and data and address wires in middle part of each sub-bank.

Fig. 3 shows the complete description of one sub-bank of the dvSRAM array and the middle part, decoder and control signal generator units. Our sub-bank design is based on Cacti suggestions and SRAM configurations that Wang et al. [14] proposed. dvSRAM array parts names come from Cacti suggested names for more clarity and simplicity.

Considering the cache structure of our system, i.e., 32 KB size, 64B lines and four-way set associativity, we need seven address bits to address a line in the array, so these seven bits and the necessary control signal bits, which we describe later, enter to the left side of Fig. 3 where the control signal generator units and address decoders can be seen. A complete description of middle part of the array is presented later in Section 2.2.2.

During an access, only one of the two sub-banks is activated; hence the numbers of bits that deal with decoder and produce word-line addresses are six; so 64 word-line addresses and 12 control signals for both sub-banks are generated there in middle part of the array and enter to the sub-bank from the left side as

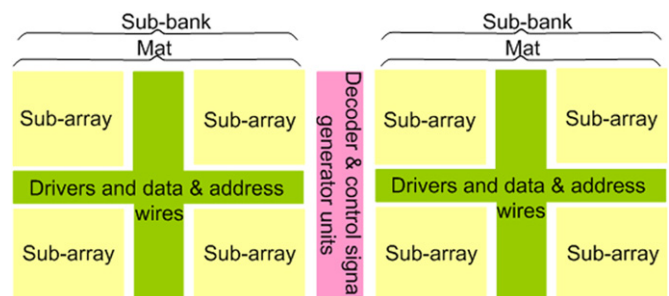


Fig. 2. dvSRAM array with two sub-banks, one mat in each of them with four identical sub-arrays. Decoder and control signal generator units are in the middle part.

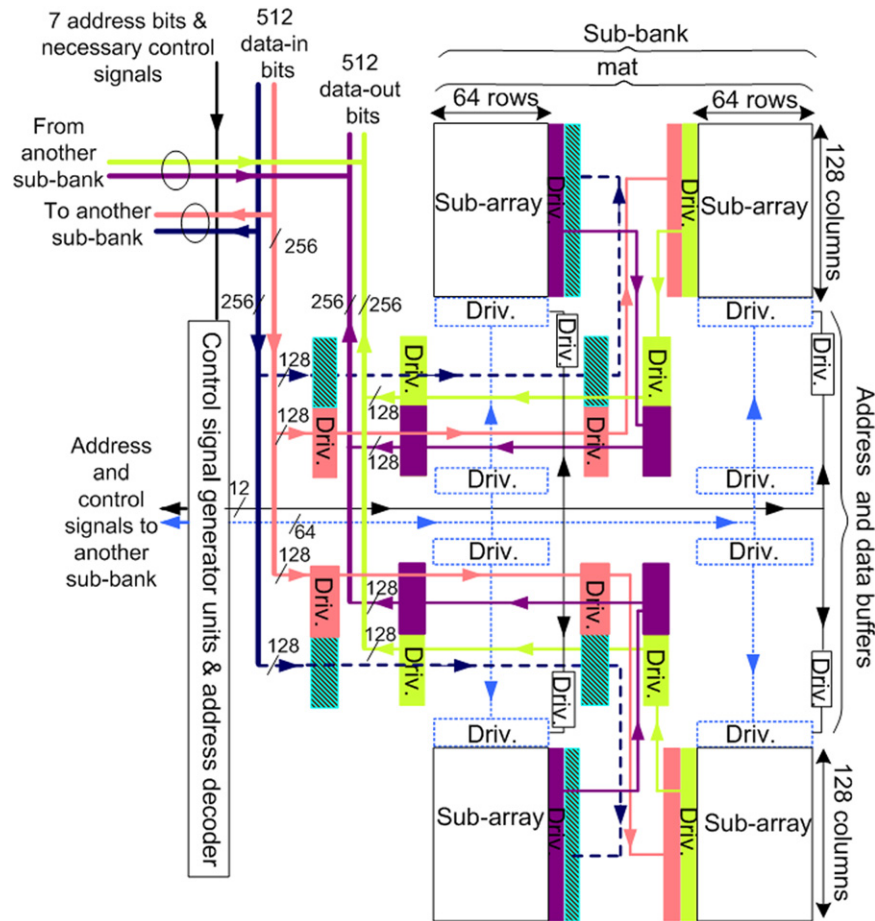


Fig. 3. A view of one dvSRAM sub-bank and the middle part of array: four identical sub-arrays, each with 64 rows and 128 columns.

can be seen in Fig. 3. 512 data-in and 512 data-out bits are routed from the up side of the sub-bank.

As mentioned before, each mat has 4 identical sub-arrays; all of them are activated during an access; therefore each sub-array holds a part of the cache line, so 128 of the 512 data bits (64B) are distributed to each sub-array. The bottom sub-arrays and the left hand side sub-arrays are mirror images of the top sub-arrays and the right hand side ones. The word-line address signals enter the upper sub-arrays through their lower sides and the lower sub-arrays through their upper sides. The data-in and data-out signals enter to the left side of the right sub-arrays and right side of the left sub-arrays. Each sub-array is composed of data-in, data-out and word-line address buffers, a 2D matrix of memory cells and associated peripheral circuitry as explained in Section 2.2.3.

We put necessary optimized drivers (chain of two series inverters) in paths to reach their related loads in the sub-arrays. 512 data-in wires enter to up side of each sub-bank. They are divided into four groups of 128 data-in wires, each for one sub-array. For the 128 data-in wires of each sub-array, there are three levels of stage drivers in the path to reach their related sub-array; two stage drivers are in the path and the last one is connected to the sub-array. Each wire and its related drivers have the same color in Fig. 3. The same method is used for data-out wires as can be seen in Fig. 3. The upper part and lower part of the sub-array is symmetric.

64 address wires re-drive twice with two levels of stage drivers in their paths to reach each sub-array as can be seen in Fig. 3. They re-drive with the first stage drivers and reach the second stage, which is connected to each sub-array. The driver

sizes are optimized to ensure that the access time for all sub-arrays is as equal as possible. As can be seen in Fig. 3, we put necessary drivers for the 12 control signals as well, using the same method employed for the address wires.

There are two important points to consider when designing the dvSRAM array. First, the wire lengths for both data and addresses from the array edge to each sub-array should be as short as possible; and second, these wire lengths for all sub-arrays of each sub-bank should be as close as possible to each other.

### 2.2.2. Middle part of the array

Fig. 4 depicts the address decoder, control signal units, and tri-state buffers that reside in the center of the dvSRAM array. To generate 64 word-line addresses from six address bits,  $A_0, \dots, A_5$ , we design a two-level decoder similar to the design of Amrutur [15]. Two 3-to-8 pre-decoders produce partially decoded products, and the main decoder generates 64 word-line addresses from their outputs. 64 word-line addresses are connected to both sub-banks via two tri-state buffer groups with enable signals. We use the highest value bit,  $A_6$  bit, as an enable signal to select the sub-bank, which is activated during the access time; so when  $A_6$  is equal to 1, the left sub-bank is activated and when  $A_6$  is equal to 0 the right one is activated.

Control signal units are responsible to generate the necessary signals for executing the proposed operations and have a synchronous flow-thru SRAM. They use one external clock to time the operations of the dvSRAM array [16]. Control signal generator unit 1 consists of some optimized inverter chains that regulate the signals to control the dvSRAM sub-array circuits such that

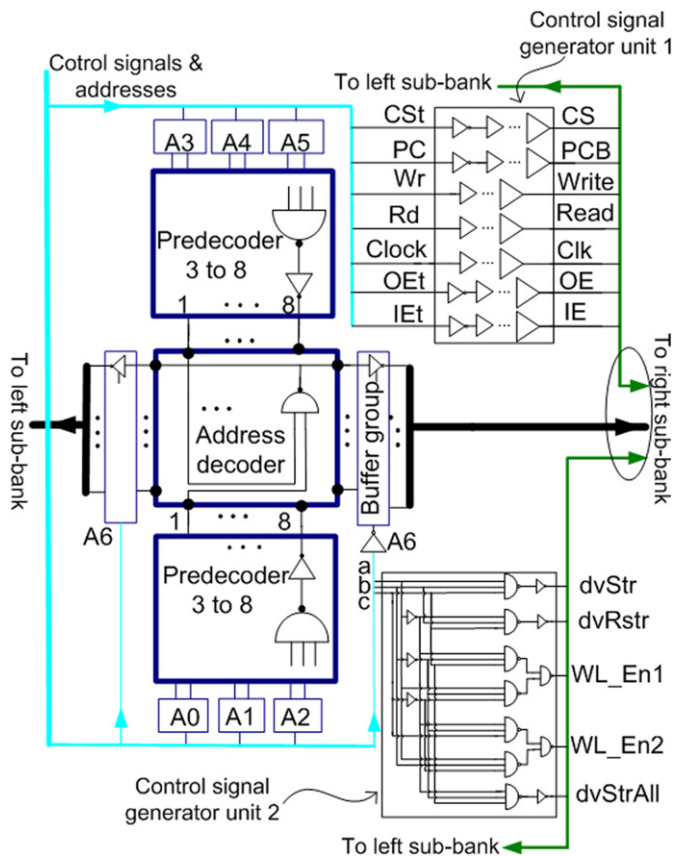


Fig. 4. Middle part of dvSRAM array: address decoders, control signal units, and tri-state buffers. Boolean functions:  $dvStr = abc$ ,  $dvRstr = \bar{a}bc$ ,  $WL\_En1 = a\bar{b}c + \bar{a}bc$ ,  $WL\_En2 = \bar{a}b\bar{c} + a\bar{b}c$ ,  $dvStrAll = \bar{a}\bar{b}c$ .

main cell operations, typical read and write can be operated correctly. Control signal generator unit 2 is a 3-to-8 decoder, generating the necessary signals for optimistic concurrency proposals:  $dvStr$ ,  $dvRstr$ ,  $WL\_En1$ ,  $WL\_En2$  and  $dvStrAll$  according to its input signal levels,  $a$ ,  $b$  and  $c$ .

### 2.2.3. Sub-array structure description

Fig. 5 shows the structure of one sub-array that consists of 2D matrix of memory cells, data-in, data-out, word-line address buffers, and associated peripheral circuitry. On the left side of Fig. 5, we can see the word-line address buffers and a set of NOR and inverter gates. As explained before, control signal generator unit 2 generates signals  $dvStr$ ,  $WL\_En1$ ,  $WL\_En2$ , and  $dvRstr$  to enable word-line address buffers. Each word-line address enters to its corresponding line via one of four buffers. For example, if we want to store the main cells of cache line 1 to their secondary cells, signals  $dvStr$  and  $WL1$  are prepared in the control signal unit 2 and decoder and then the related buffer is activated. Control signal generator unit 2 also generates  $dvStrAll$ . When  $dvStrAll$  is activated, all the main cells of whole sub-array are stored to their secondary cells simultaneously. Data-in, data-out buffers are in the bottom part of the figure. And they are typical data buffers that isolate a sub-array with interconnected wires and reduce injected noise effects.

We use typical precharge [17,8], write circuits [18], and sense amplifiers [17,8,18]; the control signal generator unit 1 generates signals to control these circuits. Each column is connected to an individual sense amplifier. The precharge circuit keeps the bit-lines ( $BL$  and  $BLB$ ) high when  $PCB$  is low. Both Read and  $dvRead$  are like read operation in typical SRAM arrays. After preparing the

appropriate word-line address in decoder and the signal Read or  $dvRead$  in control signal generator unit 1 and 2, the desired word-line is raised, turning on the pass transistors of related  $dvSRAM$  cells. Then, the bit-lines relative to the cell nodes that contain the value '0' begin discharging. At this time CS should be high and activates the related pass transistors to connect the sense amplifiers to the bit-lines. The sense amplifiers detect, which of the bit-lines are discharging and hence read the stored values [18]. The outputs of sense amplifiers are connected to their related output buffers. When OE is low, the data outputs are always tri-stated. When OE is high, they are active and the data can appear at the outputs.

Similar to a read operation, a write cycle is initiated by asserting WL. At this moment the IE is high and the data inputs can be transferred by the data input buffers to the write circuits. The write circuit is a differential stage, which is driven by two Data and Write signals. The Data and DataB signals are passed through write buffers and feed the differential inputs. Two pass transistors and the current source for the differential amplifier is controlled by the Write signal. Then, the final values and their complements are loaded onto  $BL$  and  $BLB$  [18]. The circuit behavior for Write and  $dvWrite$  is similar. Two signals  $WL\_En1$  and  $WL\_En2$  are activated simultaneously for the  $dvBWrite$  operation and Data is stored to both main and secondary cells at the same time.

Similar to Thoziyoor et al.'s [12] implementation, a layer of multiplexing can be added, before output buffers, at the outputs of sense amplifiers to select the correct way in this 4-way set associative cache; but, for simplicity, we waive this option, and the signal CS (from control signal generator unit 1, from Fig. 4) in the sense amplifiers selects the correct cache set. We show the simulation waveform of  $dvSRAM$  for all operations later.

### 2.2.4. Data and address signals distribution

Fig. 6 shows the distribution of address, data and control signals, the equivalent wire resistance and capacitance and necessary optimized drivers for one sub-array. As seen before, 64 lines of word-line addresses, 128 data-in, 128 data-out, and 12 control signal wires are routed to the sub-array from the left side of the figure. These are long wires with considerable RC delay constant and big propagation delay. For compensating the voltage drop, reflection effect and for driving big word-line capacitances, we divide the wires to shorter length and put drivers (two series inverters) at the end of each piece of wire.

As explained in Fig. 3, for each 512 data-in (data-out) wires, we put two stages of drivers in the path to reach each sub-array and one driver just in the sub-array and for the 64 word-line addresses and 12 control signal wires, we put one stage of drivers in the path to reach each sub-array and one another driver just in the sub-array. We optimize the sizes of these drivers by considering resistance and capacitance of wire sections and sub-array circuit sizes.

We can place the drivers of each stage in one line as Cacti suggests [12] but it leads to high area and un-optimized area floor-plan so we use an optimal allocation of each stage drivers that we explain as follows for both data-in and data-out wires. Instead of locating all the drivers (512 drivers) of one stage in one line, we divide them into four groups of 128 drivers (each group for one sub-array) and then we put these 128 drivers in four lines and rotate them 90 degree under clockwise as shown in Fig. 6. Setting word-line address and control signal drivers is easier and we locate them in one line at each stage. In the Figure, for simplicity, all related drivers and wires have same color.

### 2.3. Design methodology and analysis

We construct, for one array of  $dvSRAM$  and one array of a typical SRAM, HSPICE transistor level net-lists that include the

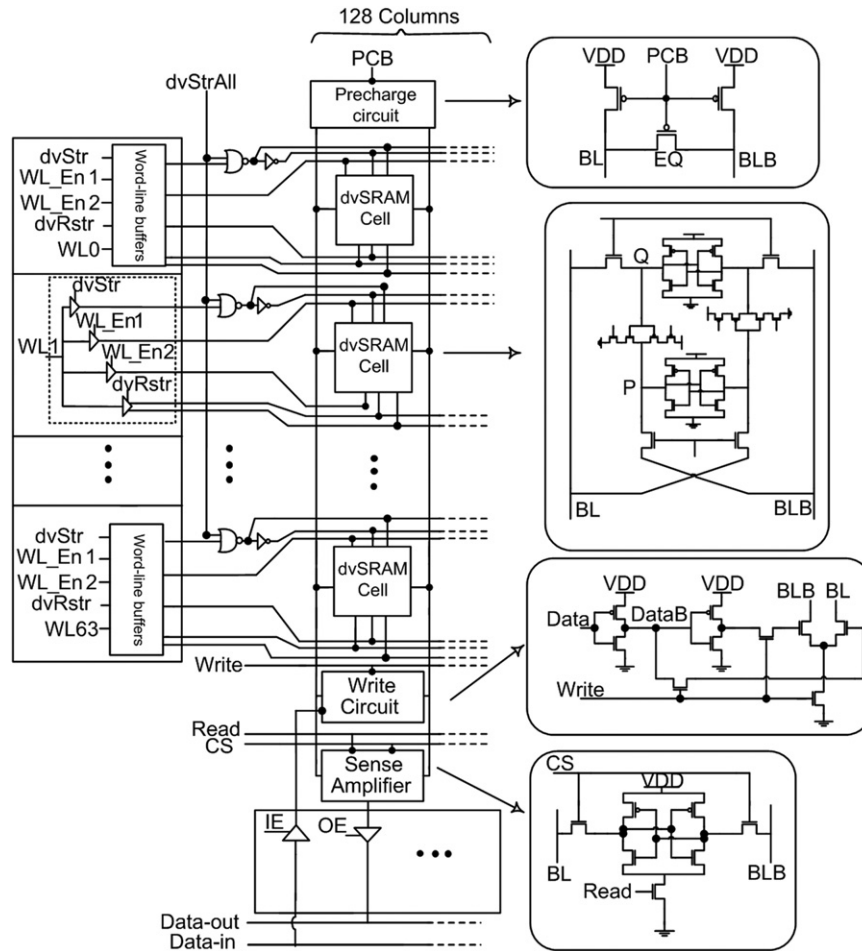


Fig. 5. Structure of one sub-array.

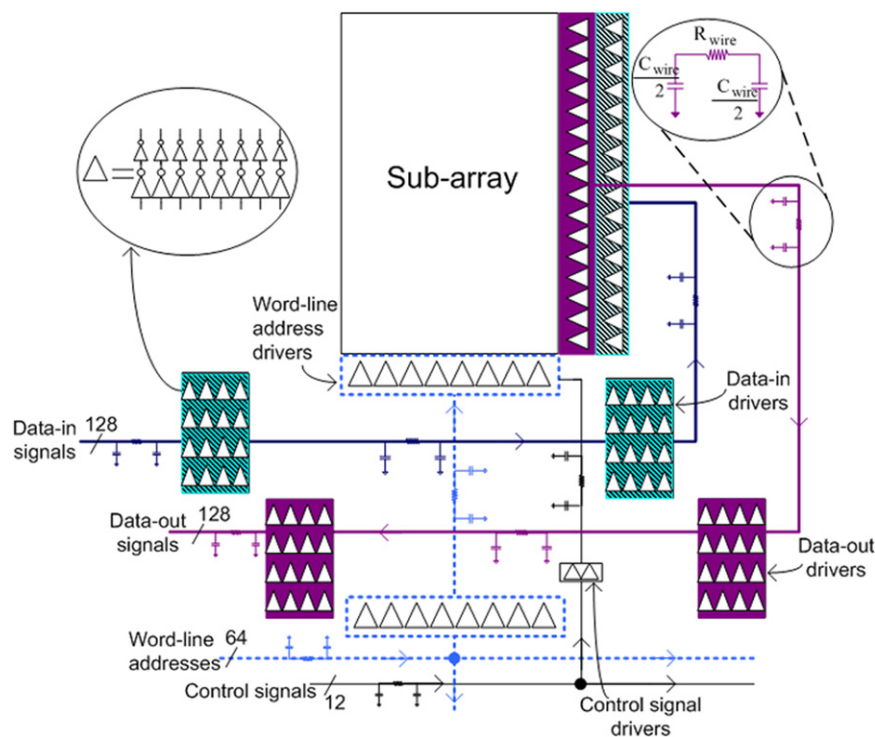


Fig. 6. Distribution of address and data drivers in a sub-array. The equivalent resistance and capacitance of each wire is shown in lumped model. Each triangle in the driver groups is representing eight drivers (two series inverters).



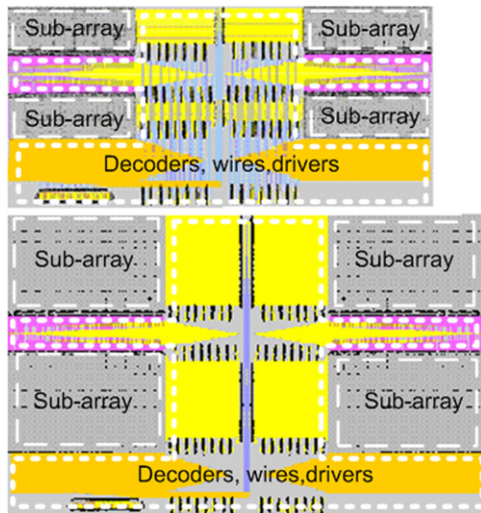


Fig. 11. Typical SRAM (top) and dvSRAM (bottom) layouts. Showing one sub-bank and address decoders.

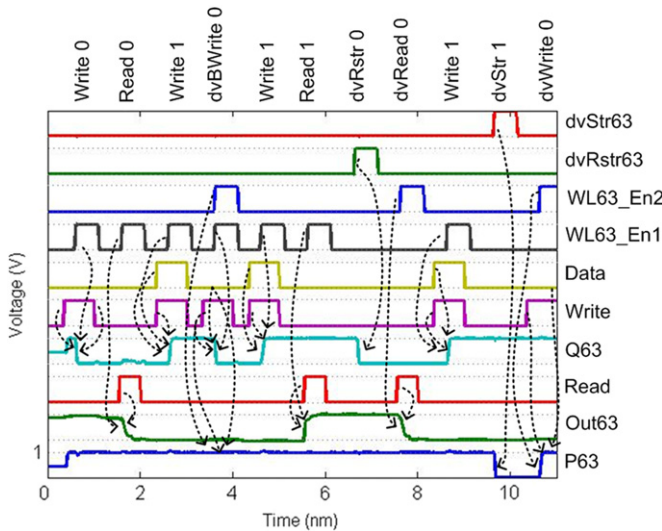


Fig. 12. Simulation waveform of dvSRAM array for a sequence of eleven operations for the first cell of the last row of one sub-array. The operations are shown on top.

Fig. 12 shows the simulation waveform for a sequence of eleven operations that take 11 ns for the first cell at the last row of one sub-array. Q63, P63 and Out 63 are internal nodes and dvStr63, dvRstr63, WL63\_En1 and WL63\_En2 are the outputs of the last word-line address buffers. Signals PC, CS, CLK, OE, IE, a, b, c are omitted because their behaviors are similar to the related signals in a typical SRAM array [8].

### 3. Use cases

In this section we discuss several use cases for the proposed dvSRAM. Moreover, a short evaluation for one of the use cases follows to show the impact of the dvSRAM.

**Speculative Multithreading (SpMT):** SpMT [1] is a concurrency mechanism that attempts to speedup sequential executions by partitioning the workload in two threads. The second thread executes the bottom half of the sequential program optimistically. Using a mechanism like the dvSRAM, each thread can use one of the independent cells, providing an easy and efficient versioning

management and a fast in-place conflict detection mechanism with state bits.

**Transactional Memory (TM):** TM [3] is a promising technique that helps with parallel program development by abstracting away the complexity of managing shared data. TM uses optimistic concurrency, assuming that conflicting data accesses will not occur; in case a conflict occurs then one or more transactions must abort, undoing all tentative data updates. This requires a multi-versioning mechanism to restore previous state. Using the dvSRAM, a partial snapshot of the past state of the system can be maintained at L1D level, leveraging a fast mechanism to restore previous state, and rely in slower mechanisms only if it is strictly necessary.

**Speculative Lock Elision (SLE):** In SLE [2] the system tries to avoid waiting in a lock when it might not be necessary by predicting potential non conflicting executions, allowing several threads to execute the same critical section optimistically. Similarly to TM, SLE has to deal with speculative updates and can benefit from the efficient dual-versioning management of the dvSRAM.

In Fig. 13 we illustrate how the dvSRAM is used in an optimistically concurrent system like the ones described above. The example shows one possible way to use the dvSRAM. Even though others might be more convenient depending on the applicability. As we can see, when an optimistic execution (speculation) starts, the system creates copies in the secondary cells using the dvStrAll operation (Fig. 13a). During the speculative section, new lines can be added to the dvSRAM upon a miss (Fig. 13b), and modifications are done in the main cell (Fig. 13c). If the system has to abort due to a conflict, it rolls-back its state using fast dvRstr operations (Fig. 13d). A comprehensive proposal of the TM use case is discussed in [5,21].

#### 3.1. Results: transactional memory

State-of-the-art HTM proposals [22] can be adapted to work with the dual-versioning cache. In particular, we evaluate in a Full-System simulation environment a state-of-the-art log-based HTM as baseline (labeled Base), and a modified version of the baseline that uses our proposed dvSRAM and logs as a fall-back mechanism only for cache lines that overflow the L1D cache (labeled DV). We selected four applications of the STAMP [23]

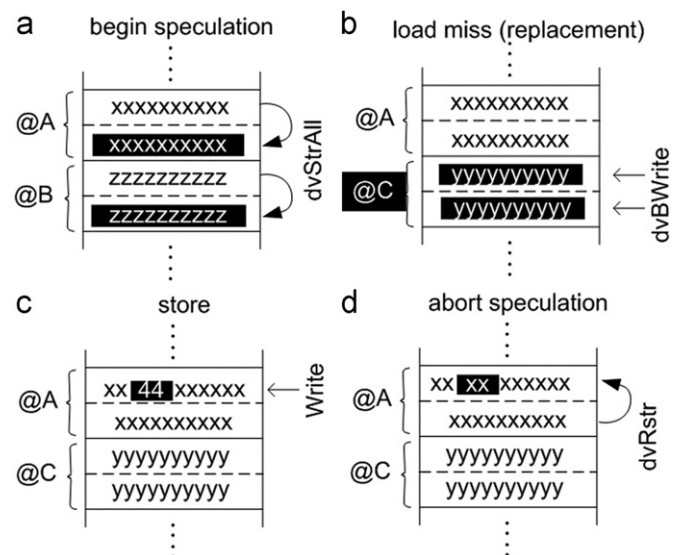
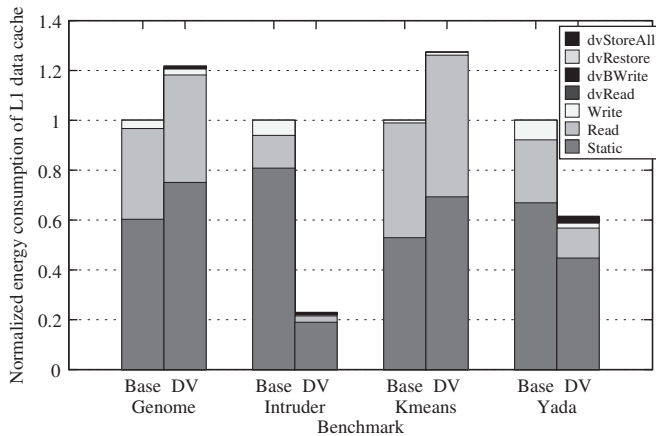


Fig. 13. Simple usage example of the dvSRAM in an optimistically concurrent system. Black back-ground indicates state changes.



**Fig. 14.** Normalized energy consumption break down of L1 data cache (Base: Baseline-HTM, DV: Dual-Versioning-HTM).

**Table 3**

Average number of each operation for all tested applications of the STAMP benchmark suite (gnome, intruder, kmeans and yada).

Read	Write	dvRead	dvBWrite	dvStr	dvRstr	dvStrAll
12912744	879772	184	431475	0	178729	20852

benchmark suite, which represent different amounts of contention, to evaluate the dvSRAM TM applicability in both performance and power consumption. In general, highly contended benchmarks scale poorly due to a larger number of aborts. This is the case of Intruder, which performs  $6.31 \times$  better when using the dvSRAM. Yada with moderate contention also shows a significant speedup of  $2.24 \times$ , while low contention benchmarks like Genome and KMeans show  $1.15 \times$  and  $1.09 \times$  speedups, respectively. This is achieved by completely avoiding the use of software logs in a large percentage of transactions (over 90% on average).

Regarding power consumption, as can be seen in Fig. 14, two applications are less energy efficient compared to a typical SRAM cache, while the other two are more energy efficient due to larger execution time speedups. In addition, for all the applications, the amount of energy spent in specific dual-versioning operations is not significant compared to the usual (Read, Write) operations and the static energy. Note that the energy results are considering only L1 energy consumption, and the L1 data cache accounts for a very small fraction of an entire processor, as discussed in Section 2.3. Thus, the energy impact considering an entire processor would be palliated.

Table 3 shows the average number of each operation for all tested applications of the STAMP benchmark suite. It is observed that the total number of accesses to the secondary cells is much lower than the total number of accesses to the main cells (4% of total accesses are accesses to the secondary cells). It can be seen that in spite of high energy consumption of dvStrAll, the number of times this operation is performed is very low compared to number of Read and Write operations.

#### 4. Related work

Ergin et al. [24] proposed similar work using a shadow cell SRAM design for checkpointed register files. In that technique, each bit-cell has a shadow-copy cell to store the temporal value, which can be recovered later. They use two cells, consisting of two back to back inverters, connected to each other using two

pass transistors and two inverters. Even when both check point and recover signals (the enable signals copy the data to the shadow-copy cell and recover it later) are inactive, pMOS transistors or nMOS transistors of these inverters are always “on” leading to power consumption increase. Moreover, if we want to modify this structure for our purpose, we should make upper inverters of cells bigger to mitigate the imbalance of this structure. While the situation is a bit better when we replace the inverters and pass transistors with each other, we still have imbalance problem which cause instability issues. All these problems lead to design a new cell with more capabilities that we can use in L1 data cache for optimistic concurrency.

In our dvSRAM structure, both cells, the main cell and the secondary cell are isolated from each other with two exchange circuits and this leads to lower static power consumption. Also, there are no unnecessary ‘on’ transistors in the active mode. We calculate and compare the static power for our dvSRAM cell and Ergin’s proposed cell to prove our discussion with HSPICE using 45 nm Predictive Technology Model (HP) for  $V_{DD}=1$  V. The static power of the dvSRAM cell is  $28.35 \mu\text{W}$  compared to the  $51.08 \mu\text{W}$  of Ergin’s cell [24].

Valero et al. proposed a SRAM-DRAM hybrid macrocell to implement first level data caches [25]. The proposed macrocell stores  $n$ -bits using one 6T SRAM cell and  $(n-1)$  1T1C DRAM cells. Both SRAM and DRAM cells are accessible externally but data can only transfer internally from SRAM cell to each DRAM cell by a pass gate transistor, which acts as a unidirectional bridge; so it is not suitable for optimistic concurrency proposals where a main cell and a secondary cell data should be accessed, modified, and moved back and forth independently.

On the other hand, Kei et al. proposed a SRAM-DRAM hybrid memory where each SRAM cell is augmented with  $n$  DRAM branches ( $2n$  1T1C DRAM cells) in a way that a value can be locally copied between the SRAM cell and one of the DRAM branches within a cell [26]. Data can be internally moved between SRAM and DRAM while only allowing access through SRAM cells, which is applicable to register files not for first level data caches.

We can use single-electron transistors and implement L1 data cache with multi-valued SRAM [27,28], which has much smaller area compared to our circuit; but this technique is too complex for our purpose because of complicated input and output circuitries for each cell.

#### 5. Conclusions

In this paper, we propose a new L1 data cache with dual-versioning SRAM cells (dvSRAM) that aims to overcome the data versioning problem present in optimistic concurrency mechanisms. We present the design details and its available operations. We calculate the power consumption, access time and design the layout. We introduce three use cases that can benefit from our proposed design. We evaluate one of them, Hardware Transactional Memory, to show our design impact in terms of performance and energy consumption. Our experiments show that the dvSRAM allows for significant performance gains with acceptable costs in power, delay and area.

#### Acknowledgments

This work is supported by the cooperation agreement between the Barcelona Supercomputing Center and Microsoft Research, by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contracts TIN2007-60625 and

TIN2008-02055-E, by the European Network of Excellence on High-Performance Embedded Architecture and Compilation (HiPEAC) and by the European Commission FP7 project VELOX (216852).

## References

- [1] V. Krishnan, J. Torrellas, A chip-multiprocessor architecture with speculative multithreading, *IEEE Transaction on Computers* 48 (9) (1999) 866–880.
- [2] R. Rajwar, J.R. Goodman, Speculative lock elision: enabling highly concurrent multithreaded execution, in: *Proceeding of 34th ACM/IEEE International Symposium on Microarchitecture (MICRO-43)*, December 2001, pp. 294–305.
- [3] T. Harris, J. Larus, R. Rajwar, *Transactional memory*, second ed., Synthesis Lectures on Computer Architecture, 2010.
- [4] A. Seyedi, A. Armejach, A. Cristal, O. Unsal, I. Hur, M. Valero, Circuit design of a dual-versioning L1 data cache for optimistic concurrency, in: *Proceeding of the 21st Great Lakes Symposium on Very Large Scale Integration (GLSVLSI'11)*, May 2011, pp. 325–330.
- [5] A. Armejach, A. Seyedi, A. Cristal, O. Unsal, I. Hur, M. Valero, Shadow HTM: using a dual-bitcell L1 data cache to improve hardware transactional memory performance, Technical report UPC-DAC-RR-2010-49, UPC, 2010.
- [6] Predictive Technology Model, <<http://ptm.asu.edu/>>.
- [7] The Electric VLSI Design System, <<http://www.staticfreesoft.com>>.
- [8] J.M. Rabaey, A. Chandrakasan, B. Nikolic, *Digital integrated circuits—a design perspective*, 2nd ed., Prentice hall, 2004.
- [9] S. Narendra, S. Borkar, V. De, D. Antoniadis, A. Chandrakasan, Scaling of stack effect and its application for leakage reduction, *International Symposium on Low Power Electronics and Design (ISLPED '01)*, August 2001, pp. 195–200.
- [10] Y. Ye, S. Borkar, V. De, A new technique for standby leakage reduction in high-performance circuits, in: *Symposium on VLSI Circuits*, June 1998, pp. 40–41.
- [11] K. Roy, S. Mukhopadhyay, H. Mahmoodi-Meimand, Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits, in: *Proceeding of the IEEE*, vol. 91, (2), February 2003, pp. 305–327.
- [12] S. Thoziyoor, N. Muralimanohar, J. Ahn, N. Jouppi, CACTI 5.1, Technical report, HPL-2008-20, HP Laboratories, Palo Alto, 2008.
- [13] Available at: <<http://quid.hpl.hp.com:9081/cacti/>>.
- [14] Yih Wang, H. Ahn, U. Bhattacharya, Z. Chen, T. Coan, F. Hamzaoglu, W.M. Hafez, C. Jan, P. Kolar, S.H. Kulkarni, J. Lin, Y. Ng, I. Post, L. Wei, Y. Zhang, K. Zhang, M. Bohr, A 1.1 GHz 12  $\mu$ A/Mb-Leakage SRAM design in 65 nm ultra-low-power CMOS technology with integrated leakage reduction for mobile applications, *IEEE Journal of Solid-State Circuits (JSSC)* 43 (1) (2008) 172–179.
- [15] B.S. Amrutur, M.A. Horowitz, Fast low-power decoders for RAMs, *IEEE Journal of Solid-State Circuits* 36 (10) (2001) 1506–1515.
- [16] *Understanding Static RAM Operation*, Technical Report IBM Application Notes, IBM, 1997.
- [17] B.S. Amrutur, *Design and analysis of fast low power SRAMs*, Ph.D. Thesis, Stanford University, 1999.
- [18] A.F. Yeknami, *Design and evaluation of a low-voltage, process-variation-tolerant SRAM cache in 90 nm CMOS technology*, Master's Thesis, Linköping University, Sweden, 2008.
- [19] S. Cosemans, W. Dehaene, F. Catthoor, A low-power embedded SRAM for wireless applications, *IEEE Journal of Solid-State Circuits* 42 (7) (2007) 1607–1617.
- [20] J. Pille, D. Wendel, O. Wagner, R. Sautter, W. Penth, T. Froehnel, S. Buettner, O. Torreiter, M. Eckert, J. Paredes, D. Hrusecky, D. Ray, M. Canada, A 32 kB 2R/1W L1 data cache in 45 nm SOI technology for the POWER7 TM processor, in: *Proceeding of IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, February 2010, pp. 344–345.
- [21] A. Armejach, A. Seyedi, A. Cristal, O. Unsal, I. Hur, M. Valero, Using a reconfigurable L1 data cache for efficient version management in hardware transactional memory, to appear in 20th International Conference on Parallel Architectures and Compilation Techniques (PACT'2011).
- [22] L. Yen, J. Bobba, M. R. Marty, K.E. Moore, H. Volos, M.D. Hill, M.M. Swift, D.A. Wood, LogTM-SE: decoupling hardware transactional memory from caches, in: *Proceedings of IEEE 13th International Symposium on High Performance Computer Architecture (HPCA '07)*, February 2007, pp. 261–272.
- [23] C. Cao Minh, J. Chung, C. Kozyrakis, K. Olukotun, STAMP: Stanford transactional applications for multi-processing, in: *Proceedings of The IEEE International Symposium on Workload Characterization*, 2008, pp. 35–46.
- [24] O. Ergin, D. Balkan, D. Ponomarev, K. Ghose, Early register deallocation mechanisms using checkpointed register files, *IEEE Transaction on Computers* 55 (9) (2006) 1153–1166.
- [25] A. Valero, J. Sahuquillo, S. Petit, V. Lorente, R. Canal, P. Lopez, J. Duato, An hybrid eDRAM/SRAM macrocell to implement first-level data caches, in: *Proceedings of IEEE/ACM International Symposium on Microarchitecture (MICRO-42)*, December 2009, pp. 213–221.
- [26] W.S. Yu, R. Huang, S.Q. Xu, S. Wang, E. Kan, G. Edward Suh, SRAM-DRAM hybrid memory with application to efficient register files in fine-grained multi-threading, in: *Proceeding of the 38th annual International Symposium on Computer Architecture (ISCA)*, 2011, pp. 247–258.
- [27] Y.S. Yu, H.W. Kye, B.N. Song, S. Kim, J. Choi, A new multi-valued static random access memory (MVSRAM) with hybrid circuit consisting of single-electron (SE) and MOSFET, in: *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2006.
- [28] Y. Yu, H.W. Kye, B.N. Song, S. Kim, J. Choi, Multi-valued static random access memory (SRAM) cell with single-electron and MOSFET hybrid circuit, *Electronics Letters* 41 (24) (2005) 1316–1317.

**Azam Seyedi** is a PhD student in Computer Architecture Department at Universitat Politècnica de Catalunya (UPC), Spain. Her current research includes low-power and high-performance VLSI design and computer architecture. She received her BS and MS in electrical engineering from Amir-Kabir University of Technology and University of Tehran, Iran.

**Adrià Armejach** is a PhD student at Universitat Politècnica de Catalunya (UPC), Spain. His research interests include multiprocessor system design, memory subsystems, and hardware support for future parallel programming models. He has an MS in computer science from Universitat Politècnica de Catalunya (UPC).

**Adrián Cristal** is a senior researcher in the Barcelona Supercomputing Center and scientific researcher in Artificial Intelligence Research Institute (IIIA-CSIC). His interests include high-performance micro-architecture, multi- and many-core chip multiprocessors, transactional memory, and programming models. He received his PhD from the Computer Architecture Department at Universitat Politècnica de Catalunya (UPC), Spain and he has a BS and an MS in computer science from the University of Buenos Aires, Argentina.

**Osman S. Unsal** is a senior researcher in the Barcelona Supercomputing Center. His interests include computer architecture, reliability, and ensuring programming productivity. He holds BS, MS, and PhD degrees in electrical and computer engineering from Istanbul Technical University, Brown University, and University of Massachusetts, Amherst, respectively.

**Ibrahim Hur** is a research scientist at Intel Corporation (he was with the Barcelona Supercomputing Center when this paper was written). His research interests include high-performance computer architecture, performance modeling, and programming models. He received his PhD from The University of Texas at Austin in Electrical and Computer Engineering, and he has a MS from Southern Methodist University, USA, and a BS from Ege University, Turkey.

**Mateo Valero** is a professor in the Computer Architecture Department at UPC and director of Barcelona Super-computer Center. His research interests include high-performance architectures. Valero has a PhD in telecommunications from UPC. He is an IEEE Fellow, an Intel Distinguished Research Fellow, and an ACM Fellow. He is a founding member of the Royal Spanish Academy of Engineering.