Register file optimization for low power using the zero detect technique.

Moisés A. Zárate S. [1], Luis Alfonso Villa Vargas [2]
Oscar Camacho Nieto [3], Osvaldo Espinosa Sosa [4]

[1,3,4] Centro de Investigación en Computación CIC-IPN
Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N
Col. Nueva Industrial Vallejo, C.P. 07738. México, DF.
TEL. 5729-600 Ext.: 56519
moizarate@correo.cic.ipn.mx, oscarc@cic.ipn.mx, espinosa@cic.ipn.mx
[2] Instituto Mexicano del Petróleo IMP
lvilla@imp.mx

Abstract. In this paper is presented an improvement in the reduction of the energy consumption for the register file of a superscalar processors with the technique of Zero Detect (detection of chains of bits equal to zeros), for which the registration is divided in chains of 32, 16, 8 y 4 *bits* (*word, integer, byte* and *nibble*), with the intention of reduce significantly the energy that wastes away when having a read or write access to the register file of this type, since the technique diminishes the transfers of chains of bits for each one of the accesses already mentioned. The carried out tests were made with the simulator Simplescalar V3.0 and with Spec95 benchmarks for each one of divisions of chains, being obtained savings in the energy consumption of up to 50%.

Words key: *Zero Detect*, Register files, Energy consumption.

1 Introduction

As it is known, the register file is a fundamental component of superscalar processors which consume a considerable amount of energy when processor needs to access to those file contents. For that reason, several techniques have been created in order to improve the performance increasing the hardware, but to cause of this, it has a higher energy consumption which is wanted to be diminished without affecting the performance [7]. One of the most important techniques on energy reduction is when having a line bit to indicate zero chains [2]. Others have sectioned those register files with multiple sub-banks in order to access every time only one bank which has an active bit line for each one of them [6]. Other techniques had a little transcendence such as separation of a register zero [1, 5], which consists on detecting single registers that contains zero values within operations that are carried out inside the functional units, which is very significant since when having an access to read the register, a lot of energy is wasted because bits are only read with zero values. It is also important to mention that others have proposed implementation of techniques that reduce multiple access that they are had when making a reading or writing to the register file [4, 8], achieving with it the energy reduction or increment in performance.

Techniques for detection of zeros in register files, caches [3], etc. are very importan since thank to them good results have been obtained for energy reduction inside superscalar processors and these techniques are susceptible of continuous improving.

2 Proposal

According to the previous necessities, we create our technique called Zero Detect which is an extension of techniques mentioned previously. It is important to mention tha from the total of registers that are used for the operations inside the functional units, can observe that 49% of them are similar to zero. As it is known, all registers that are use inside the architecture are physical, and they were born with zero value from renaming the logical registers with zero register detections were made by each reading access an writing. They can also meet groups of bits with zero value inside the registers with valu different from zero, for this reason the fundamental part of this technique is the detectio of chains of nibbles, bytes, integers and words equal to zero, for registers equal an different from zero.

Once obtained these detections of zero chains, a flag bit is placed by each one of them the activation of this bit represents a saving in the energy consumption, because only reading of it is made, avoiding the reading of all the bits of the zero chain.

2.1 The major contributor device that contains registers and detections of zer registers in readings and writings.

Inside the architecture the biggest component that offers registers to the arithmetic logic operations that are executed inside the functional units is the register file and writeback stage (fig. 1). Of these two, we only focus on the register file, which is biggest component that gives of registers, inside it only physical registers exist, of which it has been shown that the mentioned contribution is obtained from the physique ones because almost all the operations depend on the renaming of the logical, therefore, it was spoken of the bank of physical registers, that which doesn't mean that one had a division of the register file in physical and logical, it is only a short term that we will use understand that we only focus ourselves on the physical registers.

Inside all the arithmetic logical operations that were carried out with the physical register file, they were had to make two reading accesses (fig. 2), to be able to know values of the register sources and a writing access (fig. 2) to place the result of operation inside a register destination. With that mentioned, detections of zero registrations were made every time that was carried out a reading access or writing.

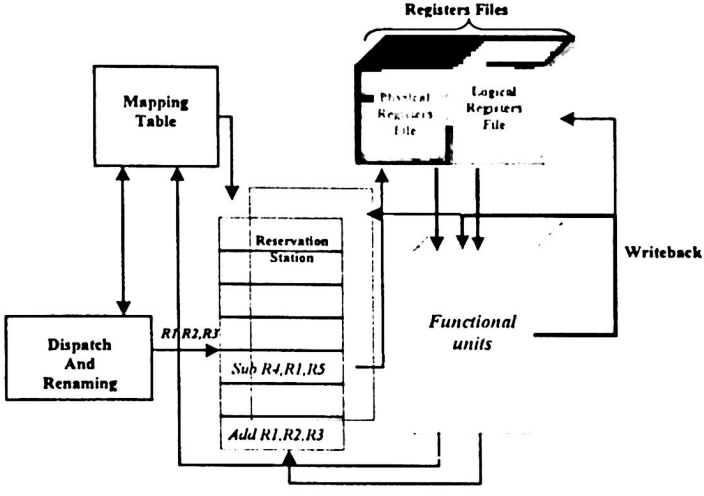

Fig. 1 Stages of superscalar processor, standing out those that contain registers.

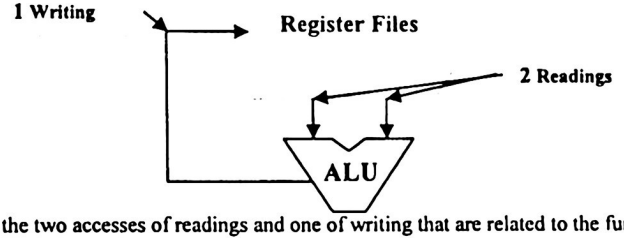

Fig. 2 It shows the two accesses of readings and one of writing that are related to the functional unit

2.2 Chain division inside the registers equal to zero and different from zero.

With the mentioned previously, once the results of all the registers are obtained with same values and different from zeros for each reading access and writings, were carried out divisions in groups of chains of nibbles, bytes, integers and words (fig. 3) inside them, which are good for us to have data but exact on the number of zero chains that use a superscalar processor. With these data is proposed the Zero Detect technique, which is explained next.

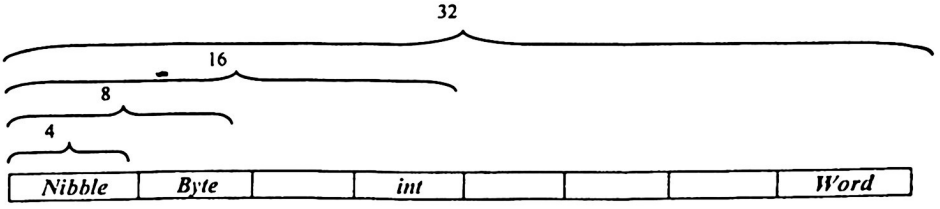

Fig. 3 Chain divisions in 4, 8, 16 and 32 bits (nibble, byte, int. and word) that were made insic each register.

2.3 Zero Detect Technique

The Zero Detect technique consists on placing a flag bit for each nibble chain, byte, and Word. The placement of this bit inside of each one of the registers increases a colum. for each chain division, like you can appreciate in the figure 4.

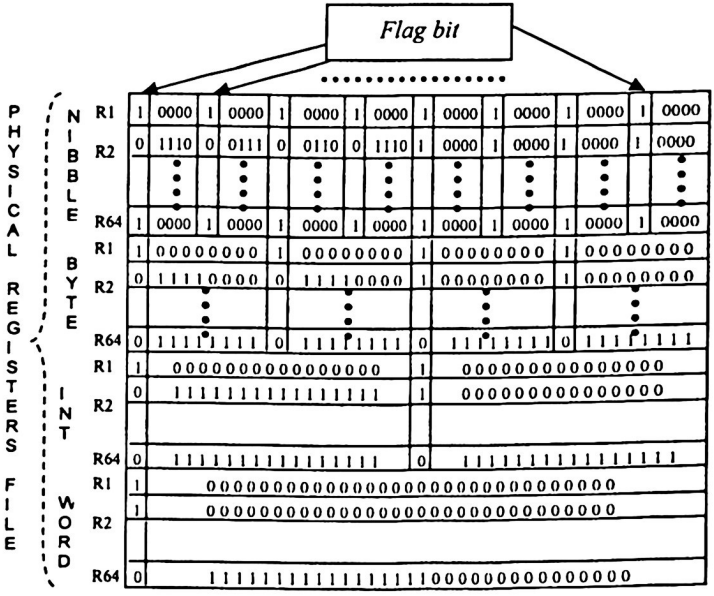

Fig. 4 It shows the Zero Detect technique applied to chains of nibbles, bytes, integers and words, which is activated when the bit flag contains a value of one, in the event of containing zero value it is not activated.

The columns containing the flag bit were the first ones in being activated to be able to know their value, which are activated when it contains a 1; this is good since alone the reading of that flag bit would be made avoiding the qualification from the whole chain to which makes reference this flag. However, if the flag bit is different from zero one would have to make the reading of the whole chain causing an extra expense of energy, because one makes the reading of the bit more the reading of the contained bits in the chains.

With the above mentioned we can observe that if the value of the bits flags were similar to 1, the saving of readings of 32 bits was obtained (words), 16 bits (integers), 8 bits (bytes) and 4 bits (nibbles), but when the value of the bit was 0 they were had to enable the chains to be able to carry out the readings of them in a second cycle, that which resulted in the reading of 33 bits (words), 17 bits (integers), 9 bits (bytes) and 5 bits (nibbles). This indicates a lost one in the saving of readings and in the time of execution due to the second cycle, but even so you speculates that the technical Zero Detect presented an energy saving.

The explanation is that at level alone transistor is wasted away energy of the activation of the 6 transistors of the flag bit and it was avoided that the 192 (words), 96 (integers), 48 (bytes) and 24 (nibbles) remaining transistors were activated.

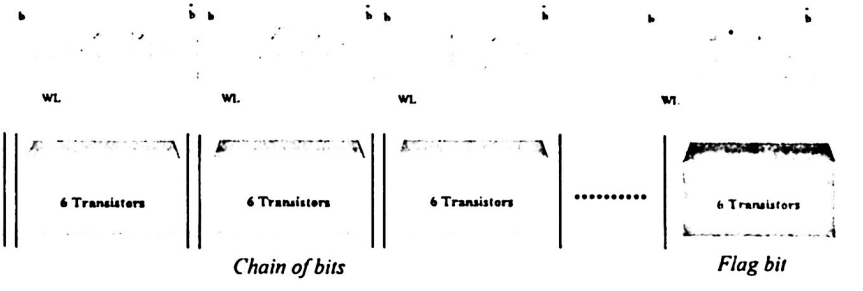

Fig. 5 It shows the number of transistors of the flag bit and of the bit's that contain the chains.

3 Methodology

To be able to know with detail the benefits that can be carried out inside a processor superscalar Alpha-21264 we use Simplescalar V3.0 simulator, because the physical development of a new processor represents a very high cost, which would have to be carried out for each experiment. The modifications that were made inside the simulator to be able to know the stage where they used most of the registers, the detections of registers zero, the divisions of chains and the implementation of the technical Zero Detect, was

proven by means of the work loads SPEC95, which are programs that were compiled the simulator.

Figure 1 is the graphic representation of the simulator with the first modification mad which consists on placing the accountants A and B inside the issue functions and accountant C in the writeback stage to know the stage where is wasted away most of registrations. In the table 1 descriptions of the benchmarks of the SPEC95 are presented.

Name of the *Benchmark*	Type of SPEC95		Description
	Integer	*Float Point*	
Compress {test}	✓		Compresses and uncompresses file in memory
li {*test*}	✓		Lisp interpreter
ijpeg {*test*}	✓		Graphic compression and decompression
Go {test}	✓		Artificial Intelligence, plays the game of Go
Vortex {*test*}	✓		A database program
M88ksim {*test*}	✓		Moto 88K Chip simulator, runs test program
gcc {test}	✓		New Version of GCC, builds SPARC code
perl {test}		✓	Manipulate strings (anagrams) and prime numbers in Perl.
applu {test}		✓	Parabolic/elliptic partial differential equations
apsi {test}		✓	Solve temp., wind, velocity and dist. of pollutants
turb3d {test}		✓	Simulate isotropic, homogeneous turbulence in cube

Table 1. Benchmarks and their description.

The benchmarks exposed in the publication are apsi, turb3d, go, applu, and ijpeg, due the space reduction that one has for the publication, however these benchmarks chosen to be representative.

4 Results

4.1 Analysis of the biggest consumer of registers inside the architecture.

The obtained results from simulations are shown in the figure 6, in the one one observes that the biggest consumer of registers inside the architecture of the process is the registers files of physical with 64%, for this reason we only focus ourselves to detection of registers zeros that they are obtained by each reading or write access in process. The writeback stage used 36% which almost represents 50% of what is used the physical register file.

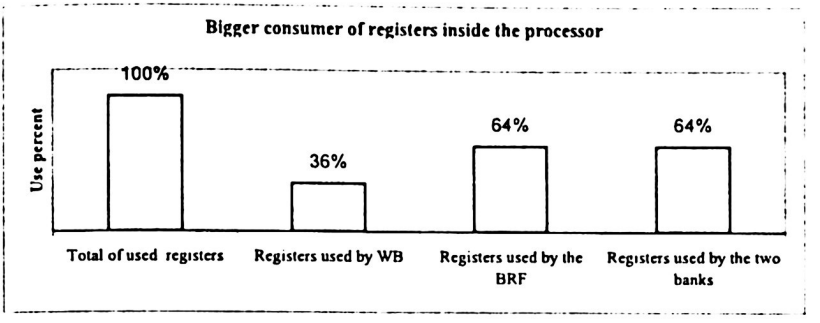

Fig. 6 It shows: 1. - Registers used by Write Back (they are all the values that are expected by a register, to be able to carry out the execution of an operation), 2. - Registers used by the registers files of physical (they are all the registers of the renaming stage that are read and they are written to liberate dependences), 3. - Registers used by the registers files of Logical (they are all the source registers inside the processor), 4. - Registers used by the banks (it indicates the total of registers in charge of inside the physical and logical register files), 5. - Total of used registers (it indicates the total number of registers that are used inside the processor).

4.2 Evaluation of zero registers detection in each reading access or writing.

The results of the detections of registers with value similar to zero in accesses of readings as well as in writings were obtained with the required simulations, which are shown in the figure 7. In her it is observed that 48% of accesses of readings were registers with value similar to zero and for the case accesses of writings one had 49% of detections of zero registers.

As it was observed, almost 50% of the registers that were used inside the architecture contained a value similar to zero, with that which several techniques could be implemented to be able to have a better performance and/or an energy saving, in our case we intended dividing the same registers and different from zeros in groups of chains of nibbles (4 bits), bytes (8 bits), integers (16 bits) and words like it is shown in the figure 3, which gives us an improvement of techniques for detections of zero registers , because it is increased the number of detections.

4.3 Evaluation of different divisions inside the zero registers.

The obtained results from different simulations are shown in figure 8, showing that 43% of readings and 42% of writings were chains of words similar to zeros. As we can observe it has a considerable quantity of chains of 32 zero bits inside the instructions to execute inside the processor. The percentages of 55% of readings and 53% of writings are also

shown, they were integers chains similar to zeros, which indicates a higher detection chains zeros in comparative with the word's chains. However it was obtained that 63% readings and 59% of writings were chains of zero bytes, giving a higher detection of zero chains as a result in comparative with words and integers. In the case of detections chains of nibbles with value similar to zero, 72% is shown for the case of readings and 67% for the case of writings. However it was obtained that 63% of readings and 59% writings were chains of zero bytes, giving a higher detection of zero chains as a result comparative with words and integers. In the case of detections of chains of nibbles with value similar to zero, 72% is shown for the case of readings and 67% for the case writings. With this it was demonstrated that when having a smaller division of word, byte and integer are obtained higher detections of chains with value similar to zero.

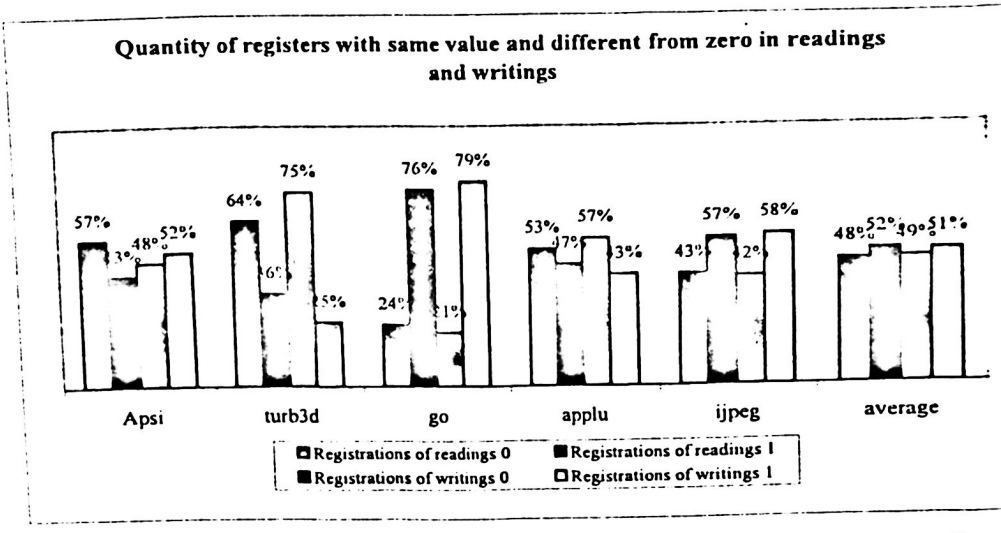

Fig. 7 Graph showing the percent of total registers with value similar to zero and one in reading and writing accesses.

Divisions of zero chains of are an important base to optimize the register files and reduce the energy consumption, due to this, the zero detect technique arises, which consented in adding a bit for each chain, like it is shown in the figure 4. This bit activated when a presented chain of zeros is detected a value similar to one, otherwise preserve its inactive state and the value that it shows are similar to zero.

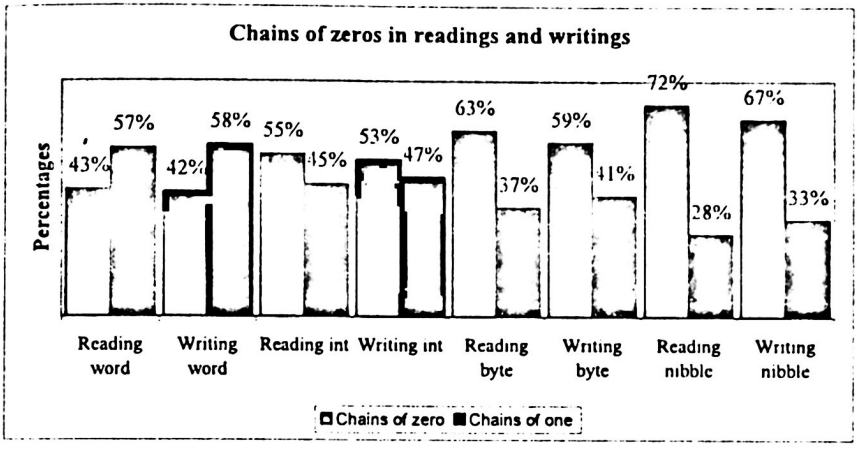

Fig. 8 Graphic showing the quantity of chains of nibbles, bytes, integers and words with value similar to zero in readings and writings.

4.4 Evaluation of the Zero Detect technique with different divisions.

As it was mentioned previously the Zero Detect technique consists on adding a flag bit to all the chains of words, integers, bytes and nibbles, for which, the results they generate are shown in the figure 9, alone they are of the activation of the flag bit. In the figure 9, it is observed that 40% of readings and 39% of writings were word's chains similar to zero. It is also shown that 49% of readings and 47% of writings were integer chains similar to zero. With the previous results you speculate that almost one had 40% of energy saving for the case of word, but for the case of integers it was almost 48%, indicating a higher optimization in the reduction of energy consumption.

For the case of bytes the graph shows that 50% readings and 46% of writings were byte's chains similar to zeros. The results showed that you almost speculates 50% of energy reduction, that which is better in comparative with word and int. Lastly in the figure 9, it is shown that 47% of readings and 42% of writings were nibble's chains similar to zero, with these values one had an optimization of little but of 45% of energy saving, that which was smaller than the int. detections and byte, but even so it was good.

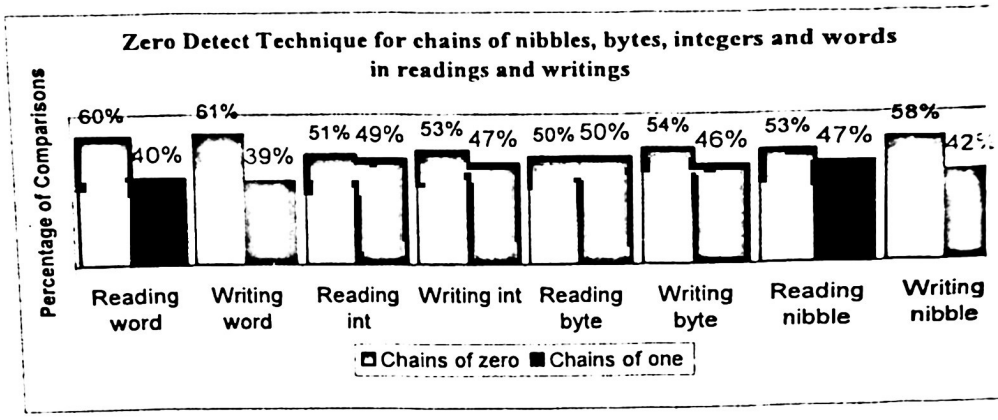

Fig. 9 Graph that shows the behaviour of Zero Detect technique for the chains of nibbles, bytes integers and words with the same and different value to zero in readings and writings.

4.5 Comparison between the renaming of a normal superscalar processor Vs. modified version with Zero Detect technique.

With the results mentioned previously it was proven that Zero Detect technique good enough, and although the performance penalties with 5% like it is shown in fig 10, we can speculate that register file was optimized in general with almost 50% of ener saving, this demonstrates that techniques for zero detections are a very important base performance and energy consumption saving.

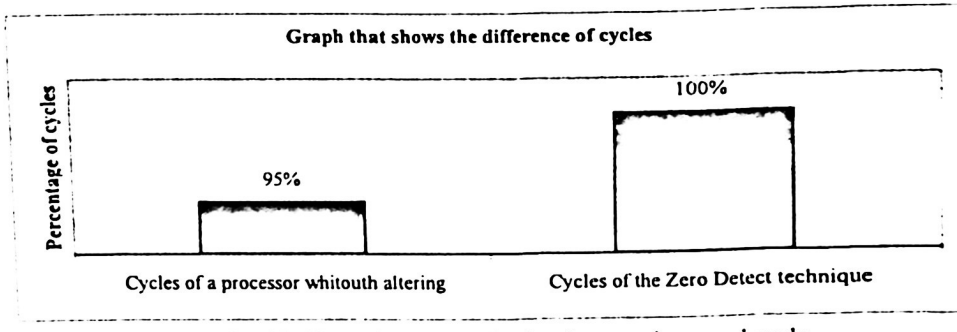

Fig. 10 Graphic illustrating the penalization due to an increased cycle.

Conclusions and future Works

All the modifications that are made inside a superscalar processor are very expensive, because the prototype doesn't work in the first experiment and this makes necessary to carry out more experiments causing a higher consumption of physical materials, without forget to take into account that sending to implement the design is very expensive. To be able to save costs in all the tests that were carried out, we already occupy one of the standard simulators called Simplescalar V3.0, which demonstrated to be efficient and able to create several models that give us an improvement in the optimization of the register bank since the result is a reflection of real tests under an energy consumption scope, therefore, we concluded that Zero Detect model created with the simulator was good since it helps to optimize the physical register file with almost 50% of energy saving, what indicates at transistor level that only it wasted away energy when being activated the 6 transistors of the flag bit in all the columns belonging to each chain, despite of penalization of 5% that we obtained in the performance.

With the percentages obtained due to optimization of energy consumption that Zero Detect technique presents, it opens the possibility of the creation of new models that allow similar consumption savings but also without having the penalization of performance. We could obtain also much precise data with software like TANNER o PSPICE in order to implement it in a Programmable Logic Device (PLD) and simulate it in power simulators.

References

[1] Jessica Hui-Chun Tseng. Energy-Efficient Register File Design. Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Master of Science in Electrial Engineering and Computer Science at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY December 1999

[2] R. Balasubramonian, S. Dwarkadas, and D.H. Albonesi. Reducing the complexity of the register file in dynamic superscalar processors. In *MICRO-34*, December 2001.

[3] Luis Villa, Michael Zhang, and Krste Asanovi´c. Dynamic zero compression for cache energy reduction. In *33rd International symposium on Microarchitecture* , Monterey, CA, December 2000

[4] A. Seznec, E. Toullec, and O. Rochecouste. Register write specialization register read specialization: A path to complexity-effective wide-issue superscalar processors. In *MICRO-35*, Istanbul, Turkey, November 2002.

[5] G. Kane and J. Heinrich. *MIPS RISC Architecture (R2000/R3000)*. Prentice Hall, 1992.

[6] J.-L. Cruz, A. Gonzalez, M. Valero, and N. P. Topham. Multiple-banked register file architectures. In *ISCA-27*, pages 316–325, 2000.

[7] Teresa Monreal Arnal. Technical Hardware to Optimize the Use of the Registrations in Processors Superescalares. University of Zaragoza, Department of Computer science and Engineering of Systems, June of the 2003.

[8] J. Tseng and K. Asanovi´c. Energy-efficient register access. In *Proc. XIII Symposium on Integrated Circuits and Systems Design*, Manaus, Brazil, September 2000.