Compiler-Driven Leakage Energy Reduction in Banked Register Files

David Atienza[1,2], Praveen Raghavan[3,4], José L. Ayala[5], Giovanni De Micheli[1], Francky Catthoor[3,4], Diederik Verkest[3,6], and Marisa Lopez-Vallejo[5]

[1] Laboratoire des Systemes Integres (LSI)/Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland
{david.atienza, giovanni.demicheli}@epfl.ch.
[2] Computer Architecture and Automation Department (DACYA)/Universidad Complutense de Madrid (UCM), Spain
datienza@dacya.ucm.es.
[3] Digital Design Technology Group (DDT)/Inter-University Micro-Electronics Center (IMEC) vzw, Heverlee, Belgium
{ragha, catthoor, verkest}@imec.be.
[4] ESAT/Katholic University of Leuven (KUL), Heverlee, Belgium
[5] Depto. de Ingenieria Electronica (DIE)/Universidad Politecnica de Madrid (UPM), Spain
{jayala, marisa}@die.upm.es.
[6] Electrical Engineering/Vrije Universiteit, Brussels, Belgium

Abstract. Tomorrow's embedded devices need to run high-resolution multimedia applications which need an enormous computational complexity with a very low energy consumption constraint. In this context, the register file is one of the key sources of power consumption and its inappropriate design and management can severely affect the performance of the system. In this paper, we present a new approach to reduce the energy of the shared register file in upcoming embedded VLIW architectures with several processing units. Energy savings up to a 60% can be obtained in the register file without any performance penalty. It is based on a set of hardware extensions and a compiler-based energy-aware register assignment algorithm that enable the de/activation of parts of the register file (i.e. sub-banks) in an independent way at run-time, which can be easily included in these embedded architectures.

1 Introduction

Current trends on communications, multimedia, networking, and other areas encourage the development of high-performance platforms that include structures to hold complex algorithms that cannot be supported by simple hardware. In this sense VLIW-like processors are the only solution that can provide a suitable performance-power trade-off for this kind of applications. Moreover, product complexity continues to increase making scalability a must of these complex architectures. Also, the time-to-market pressure provokes that a design group can no longer start from scratch. These assumptions make the use of heterogeneous multi-processor architectures the only solution to meet design goals in the

J. Vounckx, N. Azemard, and P. Maurine (Eds.): PATMOS 2006, LNCS 4148, pp. 107–116, 2006.

required time to market [1]. Several platforms have recently appeared from important semiconductor companies that confirm this tendency to multi-processor systems, e.g. ST Nomadik [2], Philips Nexperia [3] or TI OMAP [4].

On the other hand, most current systems require battery operation, which puts intense pressure on energy consumption. Given the impact that static power causes in current sub-micron technologies, as indicated by [5], energy consumption becomes a critical design metric. New embedded applications require an enormous computational performance (2 - 30$GOPS$) that need to be executed with low energy consumption demands (0.3 - 2W) for battery duration constraints [6]. Although new processors with multiple processing units can fulfill these performance figures, they consume too much power (10-100W) as outlined by [7]. Therefore, while keeping the performance results, the power consumption needs to be at least two or three orders of magnitude lower to be used in embedded environments. Within this context, methods to reduce the power consumption of the new multi-processor embedded platforms are needed.

One of the main factors that affect performance and power consumption of the new embedded platforms is the memory hierarchy. As a matter of fact, an inappropriate design of the memory subsystem and the management of the data transfers between its levels can attain up to 70% of the total power consumed in the system and up to two orders of magnitude in performance for dynamic multimedia systems [6]. Moreover, very recently it has been found that in new proposed embedded platforms with several processing elements, the shared register file plays a very important role as part of the memory subsystem and it can heavily affect the cycle time and consumes a very significant portion of the aforementioned percentage of the total energy consumed by the memory hierarchy [4]. Furthermore, it has become one of the critical processor hot-spots, specially for VLIW architectures, as [8] have shown. The main reasons are similar to those found in other layers of the memory subsystem. The register file has to support concurrent access of the several present processors and continuous exchanges of information with the L1 memory caches, therefore it is large and multi-ported. These characteristics, as occurs with memory layers on top, lead to a large increase in the power dissipation (and indirectly temperature too) of the whole system [9]. Hence, it is crucial to reduce the energy spent on it.

In this paper we introduce a new hardware approach to reduce the energy of the shared register file in upcoming embedded architectures with several VLIW processors. This work relies on a set of special hardware extensions that are controlled by the compilers of these new embedded platforms. This complete hardware/software approach enables reducing the energy consumed in the register file of forthcoming embedded architectures with multiple processing elements without incurring in performance penalties. The experimental setup is built over the CRISP (Coarse-grained Reconfigurable Instruction Set Processor) framework [10], which provides the compilation and simulation capabilities.

The remainder of the paper is organized as follows. In Section 2 we describe some related work. In Section 3 we explain the proposed architectural extensions for these new embedded platforms, while the compiler modifications

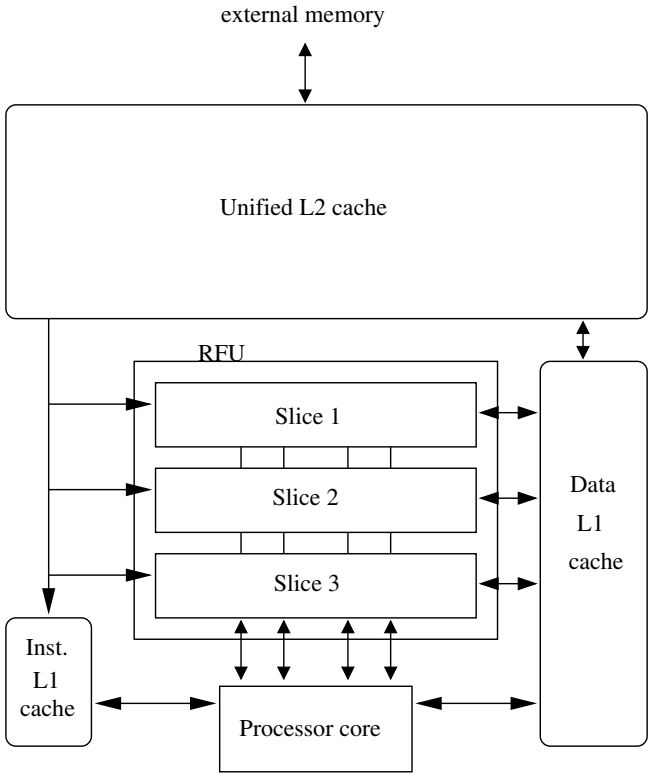

Fig. 1. CRISP: Overall emulated architecture

are explained in Section 4. Then, in Section 5, we shortly introduce the case studies and present the experimental results obtained with our proposed hardware/software approach. Finally, in Section 6 we draw our conclusions.

2 Related Work

Nowadays two major types of processing architectures have been proposed to achieve low power processing of multimedia and consumer applications. First, most forthcoming low power embedded architectures are typically customized to handle signal processing operations efficiently. Interesting domain-specific commercial DSP processors is the Coolflux Philips-PDSL [11]. Second, Application-Specific Instruction set Processors (ASIPs) have been presented as an alternative for low power embedded processing [12]. Good examples of commercially available ASIPs are Altera's NIOS [13] and Tensilica's Xtensa [14]. The academic research performed by [15] and [16] in the design of ASIPs has focused at the problem of identification and implementation of an efficient set of instruction set extensions. Although most of the work has focused on improving performance, not much work has been done specifically in the area of reducing energy

consumption. [17] presents a way to extend the instruction set based on the energy-efficiency figures of the new instructions. Even though these two types of architectures provide reduced power consumption compared to general-purpose processors, since these are both largely based on the Very Long Instruction Word (VLIW) paradigm [18], the register files are usually quite large and have several of ports (although they tend to be distributed). This leads to an energy/power wastage that is the aim of our research in this paper.

However, even though the memory hierarchy has been already largely studied, work related to the register file has started only recently. In high performance processors it can be found research devoted to defining mechanisms that decrease the energy of multi-ported register files. Regarding the hardware approaches to the problem, [19] has studied the complexity of shared register files and [20] and [21] have proposed distributed schemes and techniques to split the global microarchitecture into distributed clusters with subsets of the register file and functional units. Conversely, [22] presents other techniques that retain the idea of a centralized architecture, but the register file is split into interleaved banks, which reduces the total number of ports in each bank. In a more general context, [23] have proposed efficient voltage scaling techniques according to the application's behavior, which can efficiently reduce the overall power consumption of the system. However, in all these previous approaches it has not been studied how to apply similar techniques to multi-processor systems with shared register files by including a set of hardware extensions and power-aware compilation to control them, as we propose in this paper.

In addition, from the software viewpoint, several approaches have been proposed to alleviate the problem of the register file. In the last years, several software pipelining strategies to distribute the use of the register file, targeted at reducing memory pressure in VLIW systems, have been outlined [24]. Also, [25] and [26] have recently presented different compiler techniques, including complex register renaming, to reduce the energy spent in the register file of in-order processors. Nevertheless, such techniques were not aimed to enable the use of voltage scaling mechanisms in multi-processor environments as we introduce in the present approach.

3 Proposed Architectural Extensions

In this work we have used the compilation and simulation capabilities provided by the cycle-accurate CRISP framework [10]. It is a re-targetable compiler and simulator framework based on Trimaran [27]. The baseline architecture described by CRISP in shown in Figure 1 and consists of a selectable number of processing elements (i.e. slices) as in VLIW processors , where the compiler and architecture can be adapted to each desired DSP instruction set to be simulated. To this end, the slices of the configurable VLIW processing part are mapped onto the CRISP's Reconfigurable Functional Unit (RFU) part [10]. The RFU allows extensive customization of the VLIW processing units for a certain instruction set and an operation can be issued every clock cycle. Thus, in order to provide

enough bandwidth for the potential concurrent accesses of multiple operations, our baseline architecture includes two read and one write port of the register file, which are allocated to each slot of the VLIW processor. All functional units in one slot are connected to these three ports via a full crossbar. Regarding interconnections, the RFU part reads/writes data from/to the main shared register file. Also, CRISP includes a main processor core, which can be any type of processor, and which is used to schedule instructions that real-life VLIW systems cannot execute efficiently (e.g. control and non-parallel operations). In our experimental results (Section 5), all the real-life applications used are data-dominated and the main processor executes a very insignificant proportion of operations compared to DSP-like or loop operations (less than 5%). Thus, these figures are not considered in our reported results.

The baseline architecture described by CRISP has been extended and modified in several ways to support the power reduction mechanisms proposed in this paper. They are described in the following paragraphs of this section.

First, the register file shared among all processing elements has been split into several banks, which can be independently accessed by the processors. Then, a Dynamic Voltage Scaling (DVS) technique is applied to turn the unused banks into a low power state and thus save as much energy as possible in the system.

The register file architecture is split into independent banks and each sub-bank counts with the additional logic required to implement the DVS state. Since the low power consumption state is selected for the whole bank instead of a specific register, the overhead of the control logic is greatly minimized.

This hardware support is exploited by the compiler to power up banks of registers in the shared register file when they are needed by the several processing elements. In normal execution of the system, most banks of the shared register file are kept in a low power state thanks to our modified register assignment implemented in the compiler. In fact, only when needed, the register file banks are powered up to fulfill the register demands of the code, without performance degradation.

4 Compiler Optimizations

The complete hardware architecture depicted in the previous section has also been extended with the design of a specific compiler. The register allocation phase in the compiler of the CRISP framework was modified to exploit the new architectural feature.

The register assignment is the phase of any compiler that determines which register(s) to use for each program value selected during register allocation, while the register allocation is the phase that determines which values will be placed in registers. As a matter of fact, computer architectures (*out-of-order* processors) can destroy this first assignment by means of a hardware mechanism designed to avoid hazards, namely using *register renaming*, as was studied by [25].

Traditionally, the register assignment algorithm has been designed to choose registers from the whole pool of free registers without any other constraint. In the case of Trimaran, as many other compilers, when it tries to assign an architectural

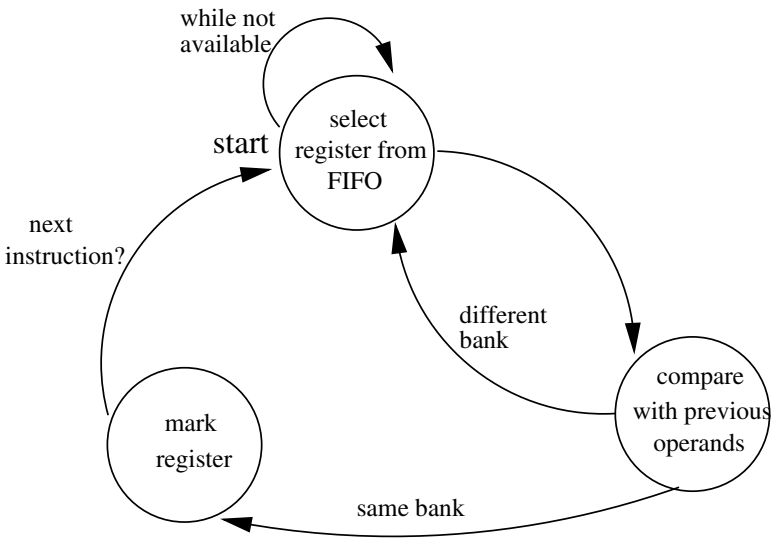

Fig. 2. Register assignment algorithm

register to the instruction operands, it retrieves the first available register from a First Input First Output (FIFO) list of free registers. In fact, the order of the registers inside the list is not representative and depends on the specific hardware architecture. Moreover, since Trimaran does not consider any restriction on assigning the registers, they are selected from the FIFO without a particular order. Therefore, the assigned registers can easily come from different register file banks if no modification to the register assignment algorithm is accomplished.

The register assignment policy we have implemented in the compiler modifies the aforementioned traditional assignment of Trimaran by promoting every operand in the instruction to the same register file bank. With this modification, most of the registers are selected from the first bank in the register file and the other banks can be kept in the low power state by turning down the voltage power supply. Instructions required for turning off the unused banks in each Basic Block was inserted by the compiler before the Basic Block.

The structure of the algorithm followed by the compiler to assign the architectural registers is shown in Figure 2. First, the first available register in the list of free registers is selected. This register is double-checked to be free and not system-reserved. Then, it is compared to the registers assigned to the other operands of the instruction. If the register file bank for the operand under assignment does not match any of the other operands of the instruction, this register is discarded and the procedure is repeated until a register belonging to the same bank is found. When the register is selected, the liveness of the register is calculated and the annotation is generated. It is important to remark that the compiler has been designed with enough flexibility to perform the register assignment algorithm with varying sizes of the register file bank, as the experimental results in Section 5 show.

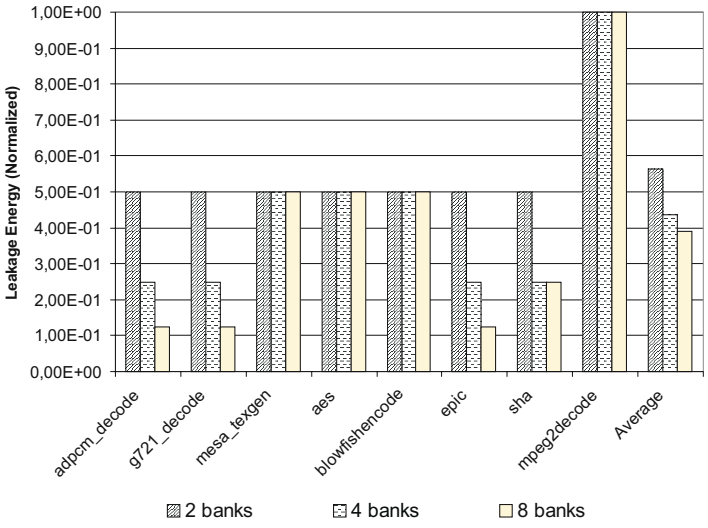

Fig. 3. Leakage Energy savings for the three configurations (normalized to non-banked register file)

Finally, when the register assignment is performed with our proposed algorithm, we can observe that in most of studied multimedia and encryption real-life applications, the registers can belong to the same bank and just few need to be selected from another register file bank (see Section 5 for more details).

5 Experimental Results

The CRISP framework detailed in Section 3 was used to simulate a VLIW processor. The processor chosen for our simulations was a 32-bit, 4-issue VLIW processor, trying to represent the forthcoming tendencies in commercial VLIW platforms for embedded multimedia applications. The register file was considered to have 12 ports (8 read and 4 write), 128 entries deep. Three cases were chosen for banking strategies: 8, 4 and 2 banks. However, the proposed architectural and compiler extensions are independent of the size of the VLIW, the ports of the register file or the number of banks. The *90*nm leakage model proposed by [28] has been used for the different register file architectures considered.

The simulator provides a trace from which the activation of each of the registers and the banks of the register file is computed. Based on the number of cycles each basic block was active and the leakage energy consumption of the bank, the net leakage energy that is consumed can be computed. The results are compared with respect to a case where no compiler modification was done. Hence, the base line case for the 4-banked register file experiment, would be a case where the compiler was not aware of such a banking.

During our first set of experiments we considered the simplest possible partition with respect to extra hardware overhead and its complexity for the

management of the register file, namely, it was divided only in 2 banks. The results obtained regarding leakage energy are depicted in Figure 3. The results shown for each benchmark using our proposed hardware/software approach have been normalized to the baseline architecture considered where all the banks of the register file are switched on during the complete execution of the program. First of all, we can observe that even with such a simple two-banked register file, we can actually reduce the leakage energy in all benchmarks (except *mpeg2decode*) by 50% as one bank can be turned to low power mode roughly during the whole execution. In the case of *mpeg2decode*, after a careful analysis of the compilation and simulation results, we could verify that this particular application puts an extremely high register pressure. Hence, all the registers are used all the time during its execution and no bank can be turned off.

Then, during the second set of experiments we explored the effect of using a more complex partitioning of the register file by considering the options of 4-banked and 8-banked register files. The results obtained for our case studies are shown in Figure 3 and Figure 3 respectively. It can be seen that, as the number of banks defined for the register file increases, the gains improve as well in most of the studied embedded applications, since they do not use a large number of simultaneous registers. Thus, more banks can be turned into low power mode. In addition, after a careful study of the results, we can observe that the type of benchmarks that benefit more from the 8-banked partitioning of the register file are the embedded multimedia applications (e.g. *mesa_texgen*, *g721_decode*, etc.). Moreover, *adpcm_decode* shows that even though it requires a significant amount of main memory to process the incoming audio (in our experiments up to 1MB), it puts very little register pressure. Hence, it enormously benefits from the increase in the number of banks considered for the register file. Conversely, the benchmarks from the cryptography domain (e.g. *blowfishencode*, *sha*, etc.) utilize more registers concurrently and do not show any improvement beyond partitioning the register file in 4 banks (see Figure 3). In the case of the *mpeg2decode* benchmark, the pressure it generates in the register file does not enable any benefit of partitioning the register file in any number of banks. Thus, there is no gain in leakage energy for any of the studied configurations.

In addition, these trends in energy improvements achieved by bank-partitioning of the register file for both types of application domains are illustrated in Figure 3. It outlines that in most of the cases the 8-banked configuration saves more energy than 2-banked and 4-banked options since it keeps a larger area of the register file in the low-power state. On the other hand, this configuration presents more overhead of extra logic needed to turn the banks into the low power state. However, this extra logic has been found to be quite simple. As a result, the gains achieved at run-time due to the low-power state of a very significant part of the register file overcome the energy overheads of the extra logic. Therefore, most of the studied benchmarks still benefit from partitioning the register file in 8 banks.

Finally, we have calculated the area overhead in the register file needed to provide the architectural extensions presented, namely, the banked-registers and the DVS technique. Our results indicate that the area overhead is only 2.16% of

the total area of the original register file respectively, which illustrates that the proposed HW/SW approach is not costly. Also, the cost of inserting the extra instructions required for the DVS was negligible, as it has to be done only for basic blocks where the execution count can be large (i.e. nested loops).

6 Conclusions

New consumer applications have recently increased in complexity and demand a very high level of performance in the next generation of low-power embedded devices. Therefore, new techniques and mechanisms that can provide solutions for an efficient mapping of these complex applications in such platforms are in great need. One of the most important factors of power consumption and performance penalty is the shared register file between all processing units. In this paper we have presented and shown in realistic examples the applicability of a new set of architectural extensions to enable the use of sub-banks in the register file, which achieves important reductions in the energy of the shared register file in upcoming embedded architectures with several VLIW processors. Our results indicate that this new integral approach enables on average a 60% reduction of the energy consumed in the register file of such forthcoming embedded architectures when they run real-life embedded multimedia, wireless network and cryptography applications without introducing performance penalties.

Acknowledgements

This work is partially supported by the Swiss FNS Research Grant 20021-109450/1, and the Spanish Government Research Grants TIN2005-05619 and TIC2003-07036. Also, this research is partially sponsored by the contract number 910467 of the Funding Program for Research of UCM-Region of Madrid.

References

1. Wayne Wolf. The Future of Multiprocessor Systems-on-Chips. In *Proceedings of DAC*, 2004.
2. ST Nomadik Multimedia Processor, 2004. http://www.st.com.
3. Philips Nexperia - highly integrated programmable system-on-chip (mpsoc), 2004. http://www.semiconductors.philips.com/products/nexperia.
4. TI's Omap platform, 2004. http://focus.ti.com/omap/docs/.
5. Nam Sung Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, and N. Vijaykrishnan. Leakage current: Moore's law meets static power. *Computer*, volume 36(12), December 2003.
6. M. Viredaz and D. Wallacha. Power evaluation of a handheld computer. *IEEE Micro*, volume 23(1), January 2003.
7. P. Bose, D. Brooks, A. uktosuno, et al., V. Zyuban, D. Albonesi, and S. Dwarkadas. Early-stage definition of LPX: A low power issue-execute processor. In *Proceedings of PACS*, November 2002.

8. A. Lambrechts, P. Raghavan, A. Leroy, M. Jayapala, T. Vander Aa, and F. Catthoor et. al. Power breakdown analysis for a heterogeneous noc platform running a video application. In *Proceedings of ASAP*, June 2005.
9. J. Abella and A. Gonzalez. On reducing register file pressure and energy in multiple-banked register files. In *Proceedings of ICCD*, 2003.
10. Pieter Op de Beeck, Francisco Barat, Murali Jayapala, and Rudy Lauwereins. Crisp: A template for reconfigurable instruction set processors. In *Proceedings of FPL*, 2001.
11. Philips PDSL. Coolflux dsp, 2005.
12. Tilman Glokler and Heinrich Meyr. *Design of Energy-Efficient Application-Specific Instruction Set Processors*. Kluwer Academic Publishers, AH Dordrecht, The Netherlands, 2002.
13. Altera. Nios embedded processor system developement, 2001.
14. R.E. Gonzalez. Xtensa: A configurable and extensible processor. In *IEEE Micro*, volume 20(2), 2002.
15. P. Biswas, V. Choudhary, K. Atasu, L. Pozzi, P. Ienne, and N. Dutt. Introduction of local memory elements in instruction set extensions. In *Proceedings of DAC*, June 2004.
16. P. Yu and T. Mitra. Characterizing embedded applications for instruction set extensible processors. In *Proceedings of DAC*, June 2004.
17. Kubilay Atasu, Laura Pozzi, and Paolo Ienne. Automatic application-specific instruction-set extensions under microarchitectural constraints. In *Proceedings of DAC*, 2003.
18. L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria, and R. Zafalon. A power modeling and estimation framework for vliw-based embedded systems. In *Proceedings of PATMOS*, Yverdon Les Bains, Switzerland, September 2001.
19. V. V. Zyuban and P. M. Kogge. The energy complexity of register files. In *Proceedings of ISLPED*, 1998.
20. A. Seznec, E. Toullec, and O. Rochecouste. Reducing register ports for higher speed and lower energy. In *Proceedings of MICRO*, 2002.
21. V. V. Zyuban and P. M. Kogge. Inherently lower-power high-performance superscalar architectures. *IEEE Transactions on Computers*, volume 50(3), March 2001.
22. I. Park, M. D. Powell, and T. N. Vijaykumar. Reducing register ports for higher speed and lower energy. In *Proceedings of MICRO*, 2002.
23. Johan Pouwelse Koen, K. Langendoen, and H. J. Sips. Application-directed voltage scaling. *IEEE Transactions on Very Large Scale Integration (TVLSI)*, volume 11(5), October 2003.
24. Cagdas Akturan and Margarida F. Jacome. Caliber: A software pipelining algorithm for clustered embedded VLIW processors. In *Proceedings of ICCAD*, 2001.
25. J. L. Ayala, M. López-Vallejo, and A. Veidenbaum. Energy-efficient register renaming in high-performance processors. In *Proceedings of WASP*, 2003.
26. J. L. Ayala and M. López-Vallejo. Improving register file banking with a power-aware unroller. In *Proceedings of PARC*, 2004.
27. Trimedia Technologies Inc. Trimaran: An infrastructure for research in instruction-level parallelism, 1999. http://www.trimaran.org.
28. P. Raghavan, A. Lambrechts, M. Jayapala, F. Catthoor, and D.Verkest. Empirical power model for register files. In *Workshop on Media and Streaming Processors (with MICRO-38)*, 2005.