Volume 41, Issue 1 January 2008 ISSN 0167-9260

ELSEVIER

INTEGRATION
the VLSI journal

SPECIAL ISSUE:
POWER AND TIMING MODELING, OPTIMIZATION AND SIMULATION
GUEST EDITORS: NADINE AZEMARD, PHILIPPE MAURINE and JOHAN VOUNCKX

CONTENTS

[Continued on outside back cover

Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com
ScienceDirect
INTEGRATION, the VLSI journal 41 (2008) 38–48
INTEGRATION the VLSI journal
www.elsevier.com/locate/vlsi

Joint hardware–software leakage minimization approach for the register file of VLIW embedded architectures ☆

David Atienza[a,b,*], Praveen Raghavan[c], José L. Ayala[d], Giovanni De Micheli[a], Francky Catthoor[c,1], Diederik Verkest[c,2], Marisa López-Vallejo[d]

[a]*LSI/EPFL, Station 14, 1015 Lausanne, Switzerland*
[b]*DACYA/UCM, Avda. Complutense s/n, 28040 Madrid, Spain*
[c]*IMEC vzw, Kapeldreef 75, 3001 Heverlee, and KULeuven Belgium*
[d]*DIE/UPM, Avda. Computense s/n, 28040 Madrid, Spain*

Received 31 January 2007; received in revised form 11 April 2007; accepted 23 April 2007

Abstract

New applications demand very high processing power when run on embedded systems. *Very Long Instruction Word* (*VLIW*) architectures have emerged as a promising alternative to provide such processing capabilities under the given energy budget. However, in this new VLIW-based architectures, the register file is a very critical contributor to the overall power consumption and new approaches have to be proposed to reduce its power while preserving system performance. In this paper, we propose a novel joint hardware–software approach that reduces the leakage energy in the register files of these embedded VLIW architectures. This approach relies upon an energy-aware register assignment method and a hardware support that creates sub-banks in the global register file that can be switched on/off at run time. Our results indicate energy savings in the register file, after considering the overhead of the added extra hardware, up to 50% for modern multimedia embedded applications without performance degradation. We illustrate this approach using real-life applications running on these processors. We also illustrate the tradeoff between the area overhead vs. the gains in the leakage energy for the different strategies.

Keywords: Register file; Low-power design; VLIW; Compiler optimization; Leakage reduction

1. Introduction

In the last years there has been an increase in computation requirements for embedded systems due to increasing complexity of new communications and multimedia standards. As a consequence, this has fostered the development of high-performance embedded platforms that can handle such high computational requirements of these recent complex algorithms, which cannot be executed in traditional RISC-like embedded processors. In addition, the continuous time-to-market pressure for consumer embedded devices has made sure that a design group can no longer perform a complete redesign each time a new product needs to be developed. Due to all these requirements, *Very Long Instruction Word* (*VLIW*) architectures have become a very attractive solution for the new consumer embedded multiprocessor architectures [1]. As a matter of fact, some platforms from the major semiconductor vendors (e.g. Philips Nexperia [2], TI OMAP [3] or ST Nomadik [4]) are already available today.

☆Part of this work has been submitted to PATMOS 2006. 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.

*Corresponding author. DACYA/UCM, Avda. Complutense s/n, 28040 Madrid, Spain. Phone: +34 91 394 76 14; fax: +34 91 394 75 27.
E-mail addresses: david.atienza@epfl.ch (D. Atienza), ragha@imec.be (P. Raghavan), jayala@die.upm.es (J.L. Ayala), giovanni.demicheli@epfl.ch (G. De Micheli), catthoor@imec.be (F. Catthoor), verkest@imec.be (D. Verkest), marisa@die.upm.es (M. López-Vallejo).
[1]Also professor at ESAT/K.U.Leuven-Belgium.
[2]Also professor at ESAT/K.U.Leuven and EE Dept/V.U. Brussels, Belgium.

0167-9260/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2007.04.004

Although VLIW-based processors have significantly improved the processing capabilities of embedded systems, they still must rely on batteries, which puts intense pressure on achieving high computational performance (2–10 Giga-OPS) at very low power (0.3–2 W). Moreover, as the impact of static power in forthcoming submicron technologies is growing [5], energy consumption becomes a critical design metric as well. In fact, the new processors with multiple processing units that can fulfill the needed performance figures consume too much power (10–100 W) [6]. Hence, new methodologies that can decrease the power consumption of current VLIW-based embedded architectures are required.

In these VLIW architectures, a very important factor is the overall power consumption and achieved performance of the final system. In fact, the correct implementation of the data transfers in the memory hierarchy and its organization can represent a variation of up to 70% of the total power consumed in the system, and up to two orders of magnitude in performance for dynamic multimedia systems [7,3]. Moreover, recent research [8] has shown that the shared register file is a key contributor of the total energy consumed in VLIW designs [9] and its correct management can dramatically affect the performance of the final system. In addition, it is a clear processor hotspot due to its size and multiple ports included to handle the continuous exchanges of information with the first level of memory caches. Due to these larger fanout requirements of the cells in the register files, the transistors in these devices are usually quite large. This makes leakage to be quite high. Clustering is one of the techniques to alleviate this problem. Our experiments show that for a TI C64x-like core,[3] the leakage energy of this clustered register file is about 60% of the whole core and about 15% of the full system on chip.[4] Therefore, reduction of leakage energy in register files is an important issue. Technology scaling also worsens these numbers as gate leakage begins to dominate as we scaled down to 65 and 45 nm devices.

While various techniques have been developed to reduce the leakage energy in the memories [10–12], there has been little research in the area of reducing the leakage energy in register files. In this paper we introduce a novel joint hardware–software method for VLIW embedded architectures that aims to reduce the leakage energy consumed in the register file. The proposed approach uses an energy-aware register assignment method in combination of few extensions of the hardware architecture. This minor hardware support provides a suitable mechanism to control the leakage energy consumed in the register file. Moreover, our results in a cycle-accurate *Coarse-grained Reconfigurable Instruction Set Processor* (*CRISP*) [13] show significant gains (up to 50%) while running realistic benchmarks from both the multimedia and wireless domains. Also, our techniques do not introduce any performance penalty and only a modest increase in silicon area.

The remainder of the paper is organized as follows: Section 2 presents an overview of related work in the area of VLIW architectures. Then, Section 3 presents the baseline VLIW architecture modeled in the CRISP framework and Section 4 explains the proposed hardware extensions for the baseline architecture. Next, Section 5 describes the energy-aware register allocation method used in our approach. Section 6 presents the experimental results while running realistic benchmarks that were considered to illustrate our approach and summarizes the obtained experimental results achieved. Finally, Section 7 summarizes the major conclusions obtained from our work and possible future research lines.

2. Related work

Nowadays two major types of processing architectures are being proposed to achieve low-power processing of multimedia and consumer applications. First of all, most of the new low-power embedded platforms (e.g. ST Nomadik by [4], Philips Nexperia by [2] and TI OMAP by [3]) fall under the Digital Signal Processing (DSP) paradigm. This type of platforms is typically customized to handle signal processing operations efficiently. Interesting domain-specific commercial DSP processors include processors like TI *C64x* series [14] and Coolflux NXP PDSL [15]. Second type includes application-specific instruction set processors (ASIPs) [16]. Good examples of commercially available ASIPs are Altera's NIOS [17], Tensilica's Xtensa [18] and STLx [19].

The academic research performed, like [20,21], has focused at the problem of identification and implementation of an efficient set of instruction set extensions. Atasu et al. [22] 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 VLIW paradigm [23], the register files are usually quite large and have several ports (although they tend to be distributed).

In all these previous options of platforms, one key element that heavily affects their deployed processing efficiency and consumed energy, apart from the register file, is the rest of the memory subsystem. Therefore, many researchers have studied different ways to optimize the memory bandwidth of the different levels of the on-chip memory hierarchy taking into account power consumption apart from performance (see [24,25] for good overviews). Also, Sahir et al. [26] and Grun et al. [27] propose in their recent works to exploit the off-chip memory bandwidth of new dynamic memories in embedded systems, such as

[3]By core, we mean the data path, including the control logic, the pipeline registers and the register file.

[4]By System on Chip we mean the core and all the L1 and L2 on-chip memories.

DRAM and SDRAM memories, by using the compiler/linker. Additional work in this field by Chang et al. [28] and De La Luz [29] has suggested how to maximize performance by distributing the data across the different banks such that as many accesses as possible can be done in parallel.

Some techniques like Drowsy Caches, ABB techniques and other leakage reduction techniques like [11] can be applied in data caches. Other circuit level approaches also exist, like [10] which reduces the leakage energy consumption in SRAMS. Some compiler techniques [30] can also be used to turn off several parts of the processor architecture based on the instruction schedule. Some other compiler techniques like [12] exploit the locality of instructions in the instruction memory to reduce the leakage energy consumption. All these techniques are complementary to the proposed approach.

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 research devoted to defining mechanisms that decrease the energy of multiported register files can be found. Regarding the hardware approaches to the problem, Zyuban and Kogge [31] have studied the complexity of shared register files and Scznec et al. [32] and Zyuban and Kogge [33] have proposed distributed schemes and techniques to split the global microarchitecture into distributed clusters with subsets of the register file and functional units. Similarly, other works like [34] have studied the benefits of multilevel register file organizations. Conversely, Park et al. [35] present 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, Koen et al. [36] have proposed efficient voltage scaling techniques according to the application's behavior, which can efficiently reduce the overall power consumption of the system. Most of these techniques are complementary to the proposed technique as they are applied at a different abstraction or on other parts of the system. To the best of our knowledge, we are not aware of techniques that reduce the leakage energy of the register file at the register allocation level with little hardware overhead.

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 [37,38]. Also, Ayala et al. [39,40] 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 multiprocessor environments, as we introduce in the present approach.

3. Initial VLIW architecture

In this work, we have used the compilation and simulation capabilities provided by the CRISP framework [13]. It is a re-targetable compiler and simulator framework based on Trimaran [41], which is cycle accurate. The baseline architecture described by CRISP consists of a selectable number of processing elements as in VLIW processors, with a coarse-grained reconfigurable logic that can be adapted to each desired DSP instruction set to be simulated. Furthermore, this framework also enables data and instruction clustering. The overall architecture of the VLIW processor used in this paper is shown in Fig. 1.

As Fig. 1 indicates, it includes a main processor core and a coarse-grained reconfigurable logic, which is divided into slices or clusters. The main processor core can be any type of processor (RISC or a multiissue VLIW) and it is included in the CRISP template architecture. Such a template is needed to execute realistic applications where the control-dominated part of the application can be in the main processor and the ILP-rich part of the application can be run parallely in the remaining units. In our simulated architecture the main processor core is a simple single-issue RISC core, but it is not relevant in our simulations since in our case studies they form an insignificant proportion of total operations compared to DSP-like or loop operations. For the loop-dominated part of the application, they are mapped on the other clusters.

The different clusters allow extensive customization of the VLIW processing units for the desired instruction set to

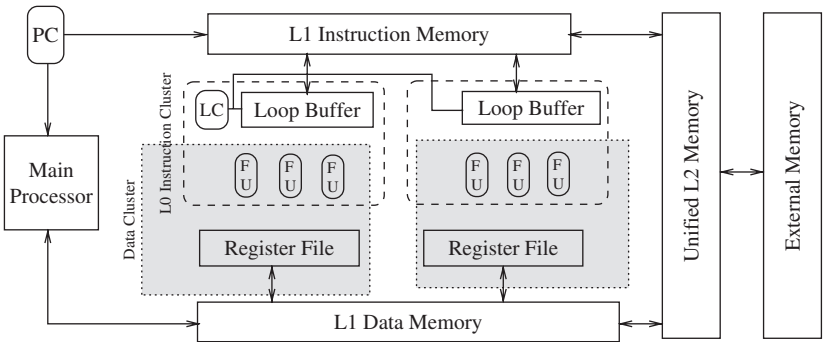


Fig. 1. Overall platform architecture.

be used in the simulated architecture within CRISP. In each cluster, an instruction can be issued every clock cycle. Regarding interconnections, different functional units (FUs) in a cluster read/write data from/to the main shared register file. In case of clustered register files, an explicit inter-cluster operation is used to transfer the data from one cluster to another. The load/store operations take place from the L1 Data memory to the register file. The RISC core and the different clusters read their instructions from the level 1 instruction cache and/or a loop buffer. The loop buffer is used when the processor is running a loop-dominated code and the instructions are fetched from the L1 Instruction Memory. For further details on the loop buffer and its operation, the reader is referred to [42]. Both types of caches (data and instruction) are connected to a unified level 2 cache, which is in turn connected to an external memory.

As we have previously mentioned, the full system is divided into clusters (e.g. in Fig. 1 two clusters are depicted). Each FU in a cluster can consist of a set of processing elements like ALUs, shifter, multiplier or memory unit. The choice of which type of processing elements are added to each of the FU slots is left to the user. For the experiments we have assumed a homogeneous set of processing elements (i.e. each FU has all types of processing elements).

In addition, each cluster is typically associated with an instruction cluster. A data cluster implies that a set of FUs use a common register file. Fig. 1 shows two data clusters, where each cluster contains three functional units. Next, an instruction cluster is formed by using a distributed instruction buffer (also called loop buffer [42,43]) across multiple-issue slots of the VLIW processor. Fig. 1 also shows two instruction clusters. In fact, all the units of one data cluster can access the data register file in that cluster. However, accessing data from another data cluster (i.e. another local register file) is relatively costly since it requires an explicit inter-cluster copy operation to transfer the data from the source data cluster to the destination data cluster. In contrast, one data cluster with all the functional units can access any data present in that register file. Therefore, in order to provide enough bandwidth for all these potential concurrent accesses, 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.

4. Register file hardware modifications

The baseline architecture described by CRISP [13] has been extended to support the power reduction mechanisms proposed in this paper. These modifications are described in the following paragraphs. Moreover, these architectural modifications enable the use of voltage scaling techniques to reduce power consumption in the register file by turning down the voltage power supply of the unused areas of this device.

First, the register file shared among all processing elements has been split into several banks, which can be independently accessed by the functional units. 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.

Turning memory devices into a low-power state has been proposed before in the literature for different objectives. In fact, previous research has focused on turning off unused memory banks (or other resources) by gating the power source. However, when the objective is a memory device, the cost of recovering the lost information could hide any power saving or, at least, represent a very significant time penalty. For example, there is an extra cost to load the data from the main memory. Moreover, when working with the register file, there is no way to recover data from memory without extra accesses to the cache. For register files, various authors have suggested that turning these unused registers into a low-power state (*drowsy* state), the power consumption can be reduced to a minimum without data loss [44].

As a result, the information in a memory cell is preserved while it is in the drowsy state. However, the data line must be restored to a high-energy mode before its contents can be accessed. One circuit technique for implementing drowsy memory devices is *Adaptive Body-Biasing with Multi-Threshold CMOS* (*ABB-MTCMOS*), where the threshold voltage is dynamically increased to yield reduction in leakage energy. This leakage reduction technique requires that the voltages of the N-well and of the power and ground supply rails are changed whenever the circuit enters or exits the drowsy mode. Since the N-well capacitance of the PMOS devices is quite significant, this increases the energy required to switch the memory cell to high-power mode and can also significantly increase the time needed to transition to/from drowsy mode. A more efficient approach to achieve the drowsy state is proposed by Flautner et al. [45], where a DVS technique is exploited to reduce static power consumption. In fact, due to short-channel effects in deep-submicron processes, leakage current is significantly reduced with voltage scaling. Thus, the combined effect of reduced leakage current and voltage yields a dramatic reduction in leakage power. This is the solution used in our approach to reduce energy consumption.

The method proposed by Flautner et al. utilizes DVS to reduce the leakage power of cache cells. By scaling the voltage of the cell to approximately 1.5 times V_{th}, the state of the memory cell can be maintained. Due to the short-channel effects in high-performance processes, the leakage current will dramatically reduce with voltage scaling. Since both voltage and current are reduced in DVS, a dramatic reduction in leakage power can be obtained. Since the capacitance of the power rail is significantly less than the capacitance of the N-wells, the transition between the two

power states occurs more quickly in the DVS scheme than the ABB-MTCMOS approach. Possible disadvantages of the drowsy circuit are increased susceptibility to noise and the variation of V_{th} across process corners. The first problem may be corrected with careful layout because the capacitive coupling of the lines is small. The second problem, variation of V_{th}, may be handled by choosing a conservative V_{DD} value.

Fig. 2 shows the modified register file cell used to support the drowsy state. As can be observed, the dual power supply is switched to low V_{DD} when the cell is in drowsy state. It is necessary to use *high-*V_{th} devices as pass transistors because the voltage on bit lines could destroy the cell contents. Before a register cell can be accessed, the power supply has to be switched to high V_{DD} to restore the contents and allow the access. An extra read_enable/write_enable gating circuit assures the memory cell is not accessed (i.e. read or written) while being in drowsy state. Furthermore, the register file architecture is split into independent banks and each subbank 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. While the above mentioned technique is one of the techniques of implementing turning off the register file banks or making them drowsy, other methods can also be used. The proposed technique can be used independent of the exact circuitry used for making the register file banks drowsy.

The previously described hardware is now exposed to the compiler to be taken advantage of. The compiler can now power up banks of registers in the register file when they are needed by the 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.

5. Energy-aware compilation

The complete hardware architecture depicted in the previous section is exploited in our approach thanks to the design of an extended energy-aware compiler. The extensions to exploit this hardware has been added to the compiler included in the CRISP framework.

The register assignment is the phase of any compiler that determines which register/s to use for each data element in the Data Flow Graph (DFG). 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 Ayala et al. [39]. Since there is no register renaming done in our architecture, this is not an issue for the proposed technique.

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 in many other compilers, when it tries to assign an architectural 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 of the underlying physical architecture. Moreover, since Trimaran does not consider any constraint 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

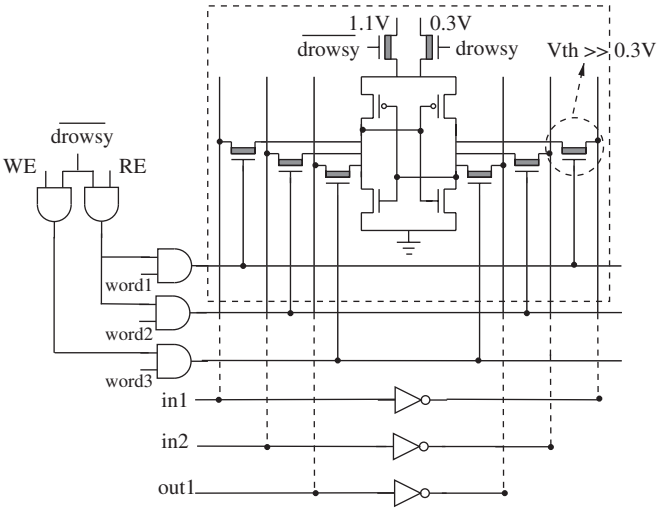


Fig. 2. Register file cell.

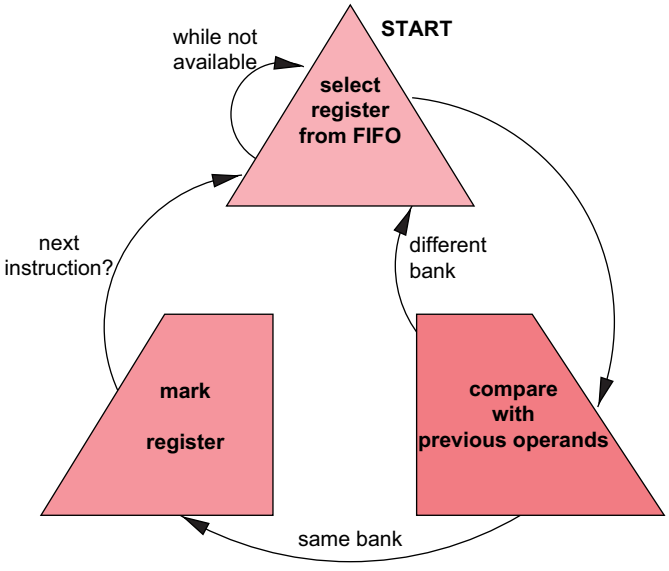


Fig. 3. Register assignment algorithm.

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. If the register pressure increase, registers may then be allocated from the next bank and so on.

The structure of the algorithm followed by the compiler to assign the architectural registers is shown in Fig. 3. 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 and other instructions in that Basic Block. 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 notice that the compiler technique is not dependent on the size of banks. This is further illustrated in Section 6.

6. Experimental results

We have applied the proposed method to several case studies that represent different modern multimedia, wireless and encryption application domains from the MediaBench benchmark suite [46] and other realistic multimedia and wireless applications. The following benchmarks were used:

- *adpcm_decode*: this benchmark performs adaptive differential pulse code demodulation, which is a very simple audio decoding algorithm particularly suited for embedded systems.
- *g721_decode*: it is a reference implementation of the CCITT (International Telegraph and Telephone Consultative Committee) for the G.721 voice decompression standard.
- *mesa_texgen*: it is an implementation of the Mesa 3D-graphics library clone of OpenGL. In our application, it is used to generate a texture mapped version of the Utah teapot. All display output functions were removed from the library due to the lack of support in CRISP for this type of I/O operation.
- *aes*: this benchmark is an implementation of the Advanced Encryption Standard (AES), also known as "Rijndael". It is a block cipher that uses a fixed-block size of 128 bits and a key size of 128, 192 or 256 bits. It has been adopted since 2001 as an encryption standard and is used worldwide today.
- *blowfishencode*: it is an implementation of the popular royalty-free block encryption algorithm, designed by the highly respected cryptographer Bruce Schneier. The blowfish encoder encrypts and decrypts in 64-bit blocks, and can use a key length of up to 448 bits.
- *epic*: this benchmark is an image compression utility that uses a bi-orthogonal critically sampled dyadic wavelet decomposition and a combined run-length/ Huffman entropy coder. It has been designed specifically for embedded systems to enable fast decoding without floating-point hardware.
- *sha*: it is an implementation of the Secure Hash Algorithm (SHA-1 Hash) for use in the Secure Hash Standard (SHS), which can be used to generate a condensed representation of a message called a message digest. It produces a 160-bit hash and there are no known attacks against it, making it very convenient for inter-communication between wireless embedded terminals.
- *mpeg2decode*: it is an implementation of the decoder of the MPEG2 standard for digital video transmission. In this case, the most important kernel inside the application is the inverse cosine transform and motion estimation, which is a highly parallel operation.
- *wcdma-transmit*: this benchmark is an implementation of wideband CMDA transmit algorithm which is one of the newer 3G standards. The data rate for the transmit is 2 Mbps. The most important part of the algorithm (from profiling) is the filter phase in the algorithm.
- *wcdma-receive*: this benchmark corresponds to the receiver end of the above wideband CMDA. The important part of this algorithm is also the filter phase and the Viterbi decoder as well.

All these benchmarks are largely loop-execution dominated, like most embedded applications. For this reason, the algorithm and our shown results focus on the registers assigned inside loops in the VLIW processing part of the system, which account for the largest part of the energy consumed (both dynamic and leakage), and do not consider the scalar registers in pure-control part of the application.

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. Further exploration of the optimal number of slots for the VLIW platform is outside the scope of this paper, but the technique proposed can be still used for wider VLIWs and also for clustered register file. In case of the clustered register file, care must be taken such that each of the clusters are handled together. The register file was considered to have 12 ports (eight read and four write), 64 entries deep. This size of register file is fairly representative of embedded systems. The TI's TMS320C64x series has such sized register files. Various banking strategies were chosen for exploration: 16, eight, four and two banks. The 90 nm leakage model proposed by Raghavan et al. [47] was used for the different register file architectures considered.

When the benchmark compilation process has finished in the CRISP framework, the percentage of utilization of the register file was obtained from the generated trace of the annotated code. Therefore, using this information obtained

during the power-aware register assignment, the resulting energy savings can be calculated for each of the banking strategies.

Let us first consider the simplest possible partition with respect to the extra hardware overhead and its complexity for the management of the register file, namely, when the register file is divided into two banks. The leakage energy savings obtained for the different banking schemes are shown in Fig. 4. 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. Firstly, we can observe that even with such a simple two-banked register file, the leakage energy of some benchmarks can be reduced by a 50% when one bank is turned to low-power mode roughly during the whole execution. In the case of some benchmarks like the *mpeg2decode*, *mesa_texgen*, *aes*, *blowfish*, after a careful analysis of the compilation and simulation results, we could verify that these application put an extremely high pressure in the register file. Hence, almost all the registers are indeed used to the fullest extent during the whole execution and no bank could be turned off.

Next we can consider the case when a larger number of banks are introduced in the register file (say four, eight and 16 banks). Fig. 4 also shows the case when four, eight and 16 banks are used. Once again the gains are normalized to the case when all the banks in the corresponding banked architecture are active. It can be seen that, as the number of banks defined for the register file increases, the gains increase 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 to low-power mode. In addition, after a careful study of the results, we can observe that the benchmarks that benefit more from the eight-banked partitioning of the register file are the embedded multimedia applications. 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 1 MB), it puts very little register pressure. Also benchmarks like *wcdma_transmit* and *wcdma_receive*, which have very high-performance constraints, require only a few number of registers. Hence, they enormously benefit 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 till 16 banks are used. In the case of the *mpeg2decode* benchmark, the register requirement is up to 59 registers and hence the gains from the proposed leakage saving strategy comes in only when 16 banks are used and one bank can be switched off.

Finally, the previously described trends of benefits achieved by bank-partitioning in the total energy consumed of the register file for both types of application domains are illustrated in Fig. 4. It shows, as has been already mentioned, that the eight-banked configuration saves more energy than the two-banked and four-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

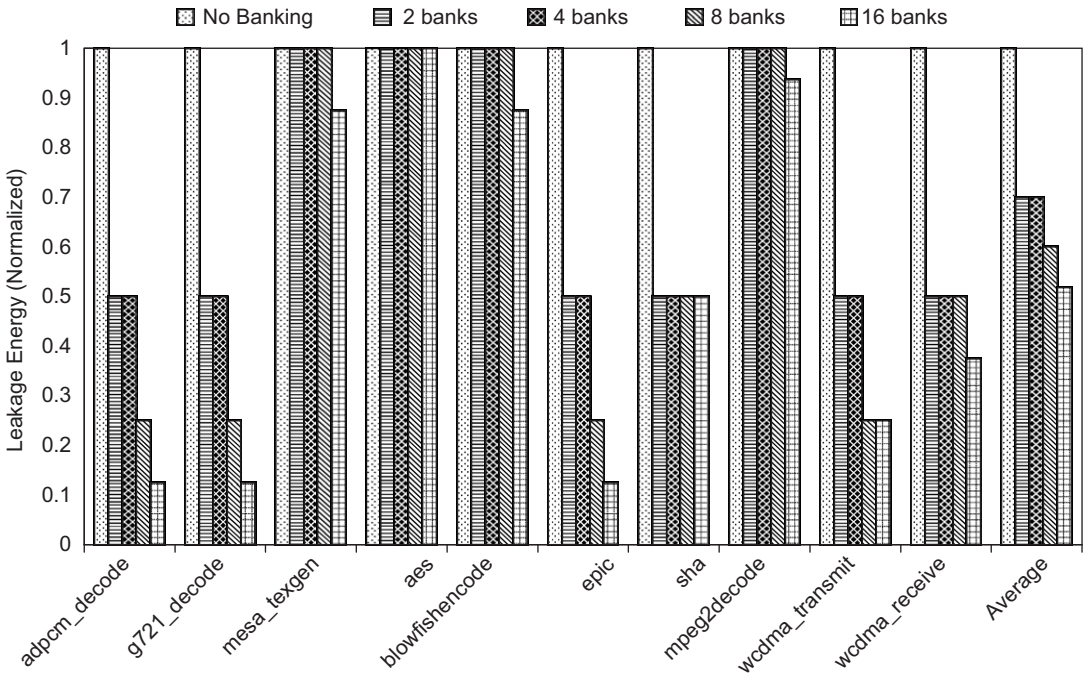


Fig. 4. Leakage energy savings for different benchmarks for various banking strategies for a 64 deep, 32 bit, eight read port, four write port register file.

achieved due to the low-power state at run time of a very significant part of the register file overcomes the energy overheads of the extra logic. Therefore, most of the studied benchmarks still benefit from partitioning the register file in eight banks.

The banking approach of the register file also comes with an extra overhead of decoders which are needed to determine which banks have to be activated for the read/write. The corresponding hardware was also synthesized and simulated in the same technology as in [47]. The

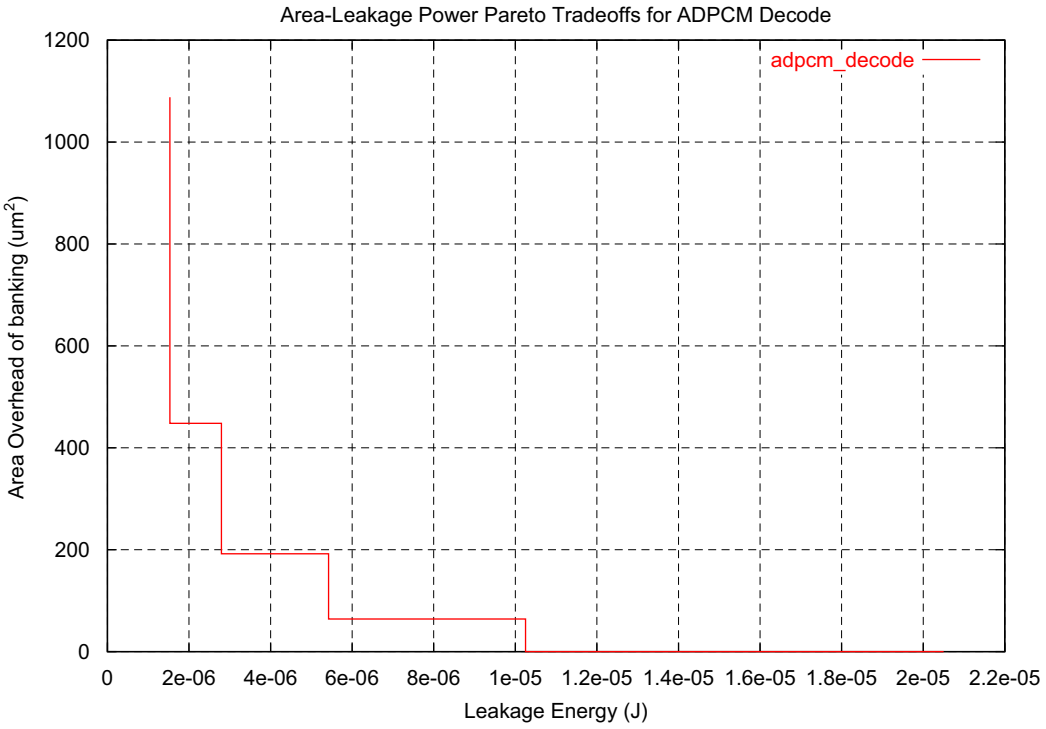


Fig. 5. Pareto tradeoff between leakage energy and area overhead for ADPCM decode.

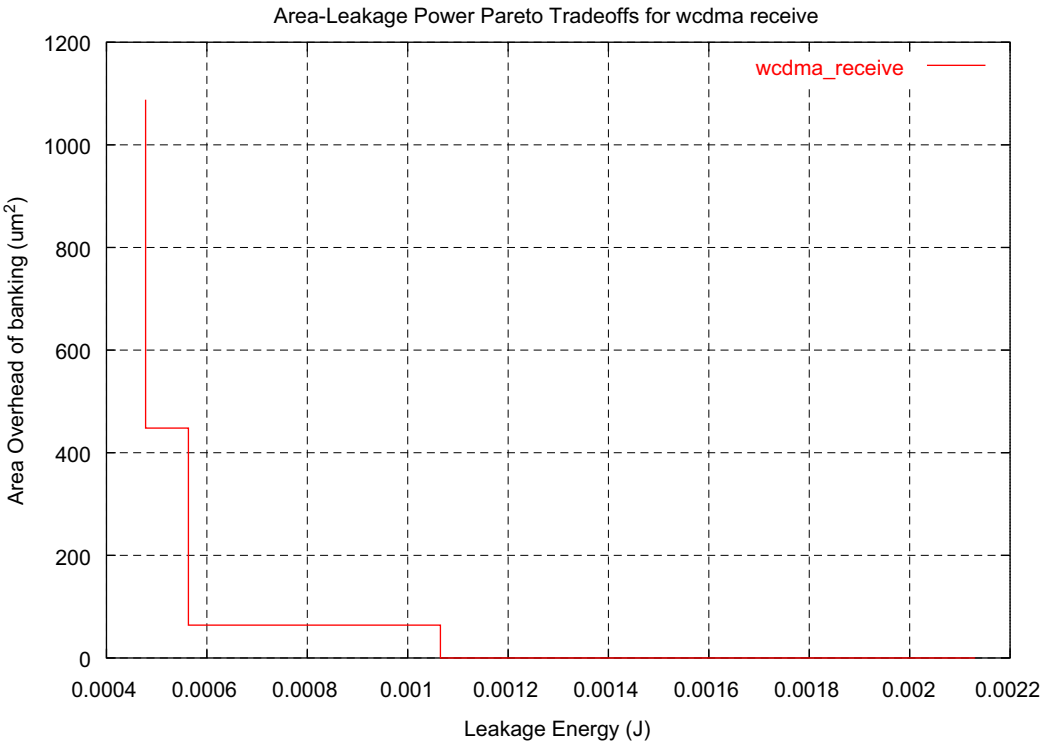


Fig. 6. Pareto tradeoff between leakage energy and area overhead for WCDMA receive.

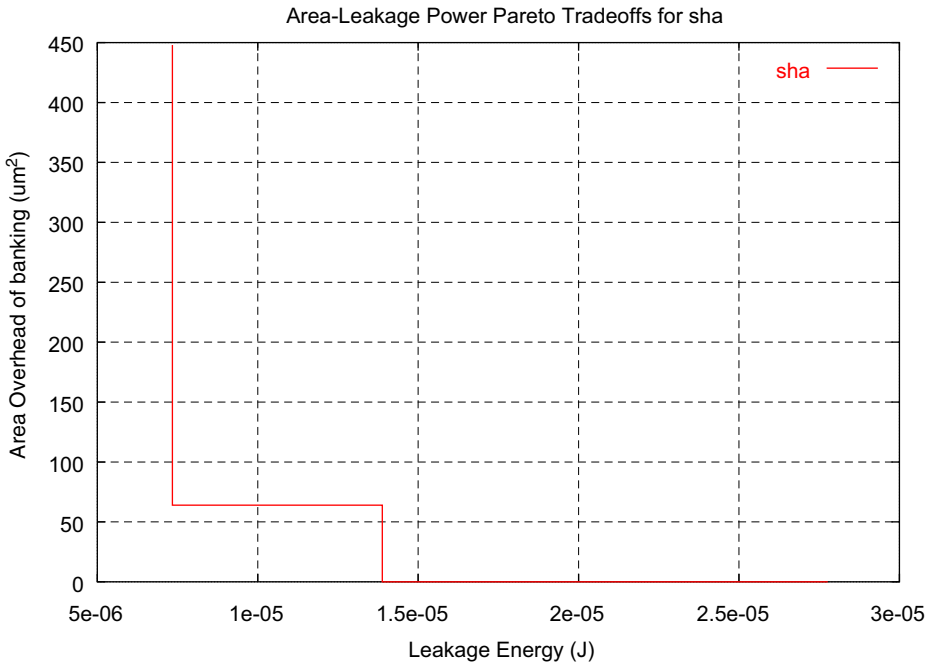


Fig. 7. Pareto tradeoff between leakage energy and area overhead for sha.

overhead in terms of energy is negligible compared to the energy consumption of the register file itself and hence is not taken for the cost. The area overhead however may be important. Also, as we increase the number of banks the decoder complexity also increases.

Based on the benchmark used the gains in the leakage energy are different. Figs. 5–7 show the leakage energy vs. area overhead pareto curve for the different banking strategies. While designing the complete system, the design can appropriately make a decision based on these tradeoffs. Each point on the curve represents a banking strategy (say no banking, two banks, four banks, etc.), when the proposed technique is applied. It can also be observed from these curves that the gains in leakage energy as we increase the number of banks follow a diminishing return.

Also there is an overhead in terms of dynamic energy for turning on and off these banks (this would include the instructions and extra hardware required for a DVS mechanism, etc). This overhead was also evaluated, on average over all the benchmarks, the dynamic energy overhead was about 5% of the total leakage energy. Therefore, this implies that such a technique would be useful provided the leakage energy gains are greater than 5% which is the case in most of the benchmarks.

7. Conclusions

Emerging consumer applications 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 presented the applicability of a new set of architectural extensions to enable the use of subbanks in the register file, which achieves significant reductions in the leakage energy of this device in VLIW processors. Our results indicate that this new integral approach enables up to a 50% reduction of the energy consumed in the register file for realistic multimedia, wireless network and cryptography applications without introducing performance penalties. We also illustrate the pareto trade-off between the area overhead vs. the leakage energy savings for the different strategies proposed.

References

[1] W. Wolf, The future of multiprocessor systems-on-chips, in: Proceedings of DAC, 2004, pp. 681–685.
[2] Philips nexperia—highly integrated programmable system-on-chip (MPSoC), 2004 ⟨http://www.semiconductors.philips.com/products/nexperia⟩.
[3] TI's omap platform, 2004 ⟨http://focus.ti.com/omap/docs/⟩.
[4] St Nomadik multimedia processor, 2004 ⟨http://www.st.com/stonline/prodpres/dedicate/proc/proc.htm ⟩.
[5] N.S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, N. Vijaykrishnan, Leakage current: Moore's law meets static power, Computer 36 (12) (2003) 65–77.
[6] P. Bose, D. Brooks, A. Uktosuno, G. Cook, K. Das, P. Emma, M. Gschwind, H. Jacobson, T. Karkhanis, P. Kudva, S. Schuster, J. Smith, V. Srinivasan, V. Zyuban, D. Albonesi, S. Dwarkadas. Early-stage definition of lpx: a low power issue-execute processor, in: Proceedings of PACS'02 (held in conjunction with HPCA), Cambridge, MA, USA, November 2002.
[7] M. Viredaz, D. Wallacha, Power evaluation of a handheld computer, IEEE Micro 23 (1) (2003) 66–74.
[8] A. Lambrechts, P. Raghavan, A. Leroy, M. Jayapala, T. Vander Aa, F. Catthoor, et al., Power breakdown analysis for a heterogeneous

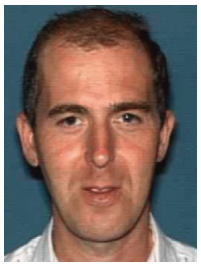
NoC platform running a video application, in: Proceedings of ASAP, June 2005.
[9] J. Abella, A. Gonzalez, On reducing register file pressure and energy in multiple-banked register files, in: Proceedings of ICCD, 2003.
[10] K. Zhang, et al., Sram design on 65-nm cmos technology with dynamic sleep transistor for leakage reduction, IEEE J. Solid State Circuits 40 (4) (2005) 895–901.
[11] S. Kaxiras, Z. Hu, M. Martonosi, Cache decay: exploiting generational behavior to reduce cache leakage power, in: Proceedings of International Symposium on Computer Architecture, July 2001.
[12] W. Zhang, J.S. Hu, V. Degalahal, M. Kandemir, N. Vijaykrishnan, M.J. Irwin, Reducing instruction cache energy consumption using a compiler-based strategy, ACM Trans. Archit. Code Optim. 1 (1) (2004) 3–33 Addresses leakage energy in instruction caches based on access history.
[13] P.O. de Beeck, F. Barat, M. Jayapala, R. Lauwereins, Crisp: a template for reconfigurable instruction set processors, in: FPL, 2001, pp. 296–305.
[14] Texas Instruments, Canada. TMS320C64x Programmer's Guide, February 2001.
[15] Philips PDSL - CoolFlux DSP, 2005.
[16] T. Glokler, H. Meyr, Design of Energy-Efficient Application-Specific Instruction Set Processors, Kluwer Academic Publishers, P.O. Box 322, 3300 AH Dordrecht, The Netherlands, 2002.
[17] Altera, Nios embedded processor system developement, 2001.
[18] R.E. Gonzalez, Xtensa: a configurable and extensible processor, in: IEEE Micro volume 20 (2), 2002.
[19] P. Faraboschi, Lx: a technology platform for customizable VLIW embedded processing, in: Proceedings of ISCA, 2000.
[20] P. Biswas, V. Choudhary, K. Atasu, L. Pozzi, P. Ienne, N. Dutt, Introduction of local memory elements in instruction set extensions, in: Proceedings of DAC, June 2004, pp. 729–734.
[21] P. Yu, T. Mitra, Characterizing embedded applications for instruction set extensible processors, in: Proceedings of DAC, June 2004, pp. 723–728.
[22] K. Atasu, L. Pozzi, P. Ienne, Automatic application-specific instruction-set extensions under microarchitectural constraints, in: Proceedings of Design Automation Conference (DAC), 2003, pp. 256–261.
[23] L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria, R. Zafalon, A power modeling and estimation framework for VLIW-based embedded systems, in: Proceedings of Power and Timing Modeling, Optimization and Simulation, Yverdon Les Bains, Switzerland, September 2001, pp. 2.1.1–2.1.10.
[24] L. Benini, G. De Micheli, System level power optimization techniques and tools, ACM Trans. Des. Autom. Embedded Syst. (TODAES) 5 (2) (2000) 115–192.
[25] P.R. Panda, F. Catthoor, N.D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, Data and memory optimizations for embedded systems, ACM Trans. Des. Autom. Embedded Syst. (TODAES) 6 (2) (2001) 142–206.
[26] M.A.R. Saghir, P. Chow, C.G. Lee, Exploiting dual data-memory banks in digital signal processors, in: ASPLOS-VII: Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems, New York, NY, USA, ACM Press, New York, pp. 234–243.
[27] P. Grun, N. Dutt, A. Nicolau. Access pattern based local memory customization for low power embedded systems, in: Proceedings of the DATE 2001 on Design, Automation and Test in Europe, Piscataway, NJ, USA, IEEE Press, New York, 2001, pp. 778–784.
[28] N. Chang, K. Kim, H.G. Lee, Cycle-accurate energy consumption measurement and analysis: case study of arm7tdmi, in: Proceedings of the 2000 International Symposium on Low Power Electronics and Design, Rapallo, Italy, 2000. ACM Press, New York, NY, USA, 2000, pp. 185–190.
[29] V. De La Luz, M. Kandemir, I. Kolcu, Automatic data migration for reducing energy consumption in multi-bank memory systems, in: DAC '02: Proceedings of the 39th conference on Design Automation, New York, NY, USA, ACM Press, 2002, pp. 213–218.
[30] W. Zhang, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, D. Duarte, Y. Tsai, Exploiting vliw schedule slacks for dynamic and leakage energy reduction, in: 34th Annual International Symposium on Microarchitecture (MICRO'01), December 2001.
[31] V.V. Zyuban, P.M. Kogge, The energy complexity of register files, in: Proceedings of ISLPED, 1998.
[32] A. Seznec, E. Toullec, O. Rochecouste, Reducing register ports for higher speed and lower energy, in: Proceedings of MICRO, 2002.
[33] V.V. Zyuban, P.M. Kogge, Inherently lower-power high-performance superscalar architectures, IEEE Trans. Comput. 50 (3) (2001) 268–285.
[34] J.L. Cruz, A. Gonzalez, M. Valero, Multiple-banked register file architectures, in: Proceedings of ISCA, 2000.
[35] I. Park, M.D. Powell, T.N. Vijaykumar, Reducing register ports for higher speed and lower energy, in: Proceedings of MICRO, 2002.
[36] J.P. Koen, K. Langendoen, H.J. Sips, Application-directed voltage scaling, IEEE Trans. Very Large Scale Integration (TVLSI) 11 (5) (2003) 812–826.
[37] C. Akturan, M.F. Jacome, Caliber: a software pipelining algorithm for clustered embedded VLIW processors, in: Proceedings of ICCAD, 2001, pp. 112–118.
[38] C. Akturan, M.F. Jacome, FDRA: a software-pipelining algorithm for embedded VLIW processors, in: Proceedings of ISSS, 2000, pp. 34–40.
[39] J.L. Ayala, A. Veidenbaum, M. López-Vallejo, Power-aware compilation for register file energy reduction, Int. J. Parallel Programming 31 (6) (2003) 449–465.
[40] J.L. Ayala, M. López-Vallejo, Improving register file banking with a power-aware unroller, in: Proceedings of PARC, 2004.
[41] Trimedia Technologies Inc. Trimaran: an infrastructure for research in instruction-level parallelism, 1999, ⟨http://www.trimaran.org⟩.
[42] M. Jayapala, F. Barat, T.V. Aa, F. Catthoor, H. Corporaal, G. Deconinck, Clustered loop buffer organization for low energy VLIW embedded processors, IEEE Trans. Comput. 54 (6) (2005) 672–683.
[43] N.P. Jouppi, Improving direct-mapped cache performance by the addition of a small fully-associative cache prefetch buffers, in: ISCA '98: 25 Year of The International Symposia on Computer Architecture (selected papers), ACM Press, New York, NY, USA, 1998, pp. 388–397.
[44] H. Hanson, M.S. Hrishikesh, V. Agarwal, S.W. Keckler, D. Burger, Static energy reduction techniques for microprocessor caches, in: International Conference on Computer Design, 2001.
[45] K. Flautner et al., Drowsy caches: simple techniques for reducing leakage power, in: Proceedings of International Symposium on Computer Architecture, 2002.
[46] C. Lee, M. Potkonjak, W.H. Mangione-Smith, Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems, in: Proceedings of International Symposium on Microarchitecture (MICRO), 1997, pp. 330–335.
[47] P. Raghavan, A. Lambrechts, M. Jayapala, F. Catthoor, D.Verkest, Empirical power model for register files, in: Workshop on Media and Streaming Processors (with MICRO-38), 2005.

David Atienza received his B.S., M.S. and Ph.D. degrees in Computer Engineering from Complutense University of Madrid (UCM) in 1999, 2001 and 2005, respectively. He currently holds a Post-Doc position at the Integrated Systems Laboratory (LSI) at EPFL. He also holds the position of invited Assistant Professor at the Computer Architecture and Automation Department (DACYA) of Complutense University of Madrid (UCM), Spain. Also, he is currently scientific counselor of long-time research at the Digital Design Technology (DDT) Group of Inter-University Micro-Electronics Center (IMEC), Leuven, Belgium.

Praveen Raghavan received his Bachelor's Degree from Regional Engineering College (REC), Trichy in India. He then received in his Masters Degree in Electrical Engineering in Arizona State University, USA. He is a researcher at IMEC vzw and also a Ph.D. candidate at KULeuven in Belgium. His interests include low power design, processor architectures and compilers.

José L. Ayala received his B.S., M.S. and Ph.D. degrees in Electronic Engineering from Technical University of Madrid (UPM), Spain, in 1999, 2001 and 2005, respectively. He currently holds a position of Assistant Professor at the Department of Electronic Engineering (IEL) of the same university. His current research activities mainly belong to the field of power and thermal estimation in processor-based systems, and the hardware/software design of embedded systems.

Giovanni De Micheli is Professor and Director of the Integrated Systems Centre at EPF Lausanne, Switzerland, and President of the Scientific Committee of CSEM, Neuchatel, Switzerland. Previously, he was Professor of Electrical Engineering at Stanford University. He holds a Nuclear Engineer degree (Politecnico di Milano, 1979), a M.S. and a Ph.D. degree in Electrical Engineering and Computer Science (University of California at Berkeley, 1980 and 1983). His research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis, hw/sw codesign and low-power design, as well as systems on heterogeneous platforms including electrical, optical, micromechanical and biological components.



Francky Catthoor received a Ph.D. in El. Eng. from the K.U.Leuven, Belgium in 1987. Since then, he has headed several research domains in the area of architectural methodologies and system synthesis for embedded multimedia and telecom applications, all within the DESICS division at IMEC, Leuven, Belgium. His current research activities mainly belong to the field of system-level exploration, with emphasis on data storage/transfer and concurrency exploitation, both in customized and programmable (parallel) instruction-set processors.

Diederik Verkest received the electrical engineering degree and the Ph.D. degree in Applied Sciences from the Katholieke Universiteit Leuven (Belgium) in 1987 and 1994, respectively. In the period 1987–1994 he worked as a research assistant in the domain of formal design and verification methodologies. Since 1994, he has been working in the VLSI design methodology group of the IMEC laboratory (Leuven, Belgium) on system design and hardware/software co-design as part of the team that developed the CoWare environment. From 1997 to 2001, he headed research in the Embedded System design technology group in IMEC. Since 2001, he is the principal scientist in the T-ReCS group where he is responsible for the research strategy related to re-configurable system design. Diederik Verkest is guest professor at the University of Brussels (VUB).

Marisa López-Vallejo received the M.S. and Ph.D. degrees in Electronic Engineering from the Technical University of Madrid in 1992 and 1999, respectively. She joined the Faculty of the Technical University of Madrid in 1993 and is currently an Associate Professor in the Department of Electronic Engineering. Her research interests include VLSI design for communication systems and CAD tools for Hardware–Software Codesign, with particular emphasis on power optimization for embedded systems.