# Hybrid Functional and Instruction Level Power Modeling for Embedded Processors

Holger Blume[1], Daniel Becker[1], Martin Botteck[2], Jörg Brakensiek[2], and Tobias G. Noll[1]

[1] Chair for Electrical Engineering and Computer Systems
RWTH Aachen University, Schinkelstr. 2, 52062 Aachen, Germany
[2] Nokia Research Center
Meesmannstr. 103, 44807 Bochum, Germany
{blume, becker, tgn}@eecs.rwth-aachen.de,
{martin.botteck, jorg.brakensiek}@nokia.com

**Abstract.** In this contribution the concept of Functional-Level Power Analysis (FLPA) for power estimation of programmable processors is extended in order to model even embedded general purpose processors. The basic FLPA approach is based on the separation of the processor architecture into functional blocks like e.g. processing unit, clock network, internal memory etc. The power consumption of these blocks is described by parameterized arithmetic models. By application of a parser based automated analysis of assembler codes the input parameters of the arithmetic functions like e.g. the achieved degree of parallelism or the kind and number of memory accesses can be computed. For modeling an embedded general purpose processor (here, an ARM940T) the basic FLPA modeling concept had to be extended to a so-called hybrid functional level and instruction level model in order to achieve a good modeling accuracy. The approach is exemplarily demonstrated and evaluated applying a variety of basic digital signal processing tasks ranging from basic filters to complete audio decoders. Estimated power figures for the inspected tasks are compared to physically measured values. A resulting maximum estimation error of less than 8 % is achieved.

## 1 Introduction

In the course of increasing complexity of digital signal processing applications, especially in the field of mobile applications, low power techniques are of crucial importance. Therefore, it is desirable to estimate the power consumption of a system at a very early stage in the design flow. By this means, it is possible to predict whether a system will meet a certain power budget before it is physically implemented. Necessary changes in the system or the underlying architecture will then be much less time and money consuming, because no physical implementation is required to determine its power dissipation.

Like any other architecture block the power consumption of a processor depends on several factors like the switching activity of the input data, the clock frequency and of course the executed task itself. Besides these dependencies

S. Vassiliadis et al. (Eds.): SAMOS 2006, LNCS 4017, pp. 216–226, 2006.

there are many more processor-specific factors like the type and rate of memory accesses, the usage of specific architecture elements, different compiler settings, pipeline stalls and cache misses but also different programming styles or algorithmic alternatives which all strongly influence the power consumption of a task that is executed on a processor.

For this reason it is desirable to consider methodologies for power estimation that cover all significant influencing factors and provide a sufficient accuracy at moderate complexity. Such a methodology is presented in this paper and verified using exemplary vehicles. The paper is organized as follows: Chapter 2 shortly discusses several existing power estimation techniques in terms of their applicability to modern processor kernels. The following chapter describes the so-called Functional-Level Power Analysis (FLPA) approach in detail. Chapter 4 describes the required basics of the ARM940T general purpose processor architecture which is exemplarily modeled here. Chapter 5 explains the exemplary modeling of this processor architecture and works out the need for a hybrid FLPA/ILPA approach. A benchmarking of the hybrid FLPA/ILPA model is performed in chapter 6. Finally, a conclusion of the paper is given in chapter 7.

## 2 Classical Approaches for Power Estimation

One possible straight forward power estimation approach on processors is the so-called Physical-Level Power Analysis methodology. This approach is based on the analysis of the switching activity of all circuit nodes of the processor architecture. The requirement of this methodology is the availability of a detailed description of the processor architecture on transistor level, which is rarely given for modern processors and even more severe results in an extremely high computational effort. Architectural-Level approaches like [1] reduce this computational effort by abstracted modeling typical architecture elements like registers, functional units or load/store queues. Therefore, these methodologies can be mainly found in the development of high volume products like e.g. microprocessors. Due to their extremely high computational effort they are not suited to evaluate the power consumption of complete digital signal processing tasks performed on a processor in practical use with acceptable computation times.

Another possibility for power estimation for processors is the so-called Instruction-Level Power Analysis [2]. By means of low level simulations or physical measurements the energy consumption of each instruction out of the instruction set of a processor is determined. By analysis of the assembler code of a task it is possible to estimate the specific power consumption of this program performed on a certain processor. The advantage of this approach is to cope with so-called inter-instruction effects. In general, the energy consumption of a processor instruction depends on the previously executed instructions, what can be explained by means of Fig. 1 and Fig. 2.

At a certain stage of a processors pipeline, instruction words are transferred from the program cache into a register in the processor core for further processing. Fig. 1 shows the situation that an ADD instruction word replaces a MUL

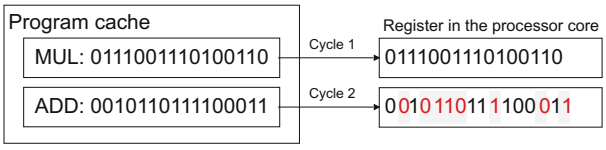


**Fig. 1.** Sequential execution of two *different* processor instructions

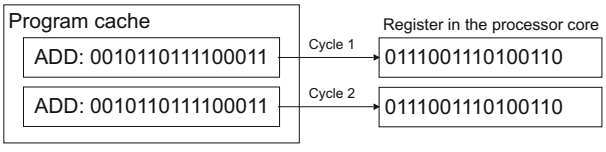


**Fig. 2.** Sequential execution of two *identical* processor instructions

instruction word in cycle 2. The numbers shaded by gray boxes show the bits in the register that switch their state in this case. In this example a Hamming distance (number of different bits of these two instruction words) of eight ($H_d$=8) is resulting. Of course the sequence of two identical instructions causes no switching activity ($H_d$=0), (Fig. 2). Effects like this occur in many stages of a processors pipeline and as a result of these effects the energy consumption of an instruction obviously depends on the previously executed instruction. The Instruction-Level Power Analysis methodology allows to cover such inter-instruction effects by measuring the energy consumption of groups of processor instructions, but the huge number of possible combinations makes this approach very complex. The effort will even grow, if Very-Long-Instruction-Word (VLIW) architectures shall be modeled due to their increasing word length and their ability to issue several operations in parallel.

A more attractive approach for power estimation is the Functional-Level Power Analysis (FLPA) methodology. This methodology has been introduced in [3] and was first applied in [4] to a digital signal processor. Furthermore, in [5] or [6] it could be shown that a good estimation accuracy can be achieved. Here, an extension of this methodology is presented in order to model even processor cores which feature a strong dependency of the corresponding power consumption on the performed instruction. According to this, a so-called hybrid FLPA/ILPA model is elaborated which advantageously combines the low modeling and computational effort of an FLPA model and the higher accuracy of an ILPA model.

## 3 Functional Level Power Analysis (FLPA)

The basic principle of the FLPA methodology is depicted in Fig. 3. In a first step the processor architecture is divided into functional blocks like fetch unit, processing unit, internal memory etc. By means of simulations or measurements it is possible to find an arithmetic model for each block that determines its power consumption in dependency of certain parameters. These parameters are

for example the degree of parallelism, the access rate of the internal memory or the clock frequency. Most of these parameters can be automatically determined by a parser which analyzes the assembler file of a program code. The total power consumption is then given as the sum of the power consumption of each functional block.

The left side of Fig. 3 depicts the process of extracting parameters from a program implementing a task. In this second step it is possible after compilation to extract the task parameters from the assembler code. Further parameters can be derived from a single execution of the program (e.g. the number of required clock cycles). These parameters are the input values for the previously determined arithmetic models. Thus, an estimation of the power consumption of a given task can be computed. This approach is applicable to all kind of processor architectures without detailed knowledge of the processor architecture. Furthermore, FLPA modeling only requires moderate time effort.

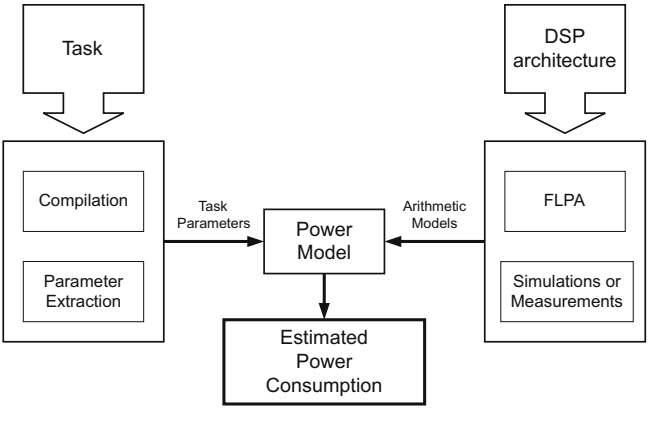


**Fig. 3.** The basic FLPA principle

## 4 Architecture of the ARM940T Processor

The 32 bit general purpose processor ARM940T is targeted for mobile applications, e.g. smart phones. Both hardware and instruction set architecture are based on the ARM v4T reference architecture (see e.g. [7]).

Like the ARM v4T, the ARM940T processor is based on a Harvard-architecture. The ARM940T consists of an ARM9TDMI Reduced Instruction Set Computer (RISC) -processor core and separate instruction and data caches (4KByte each). Additionally, the ARM940T provides interfaces for coprocessors and the special Advanced Microcontroller Bus Architecture (AMBA) interface, as well as the system configuration coprocessor (CP15), which is used to control e.g. the memory protection unit [8]. A block diagram of the ARM940T architecture is depicted in Fig. 4.

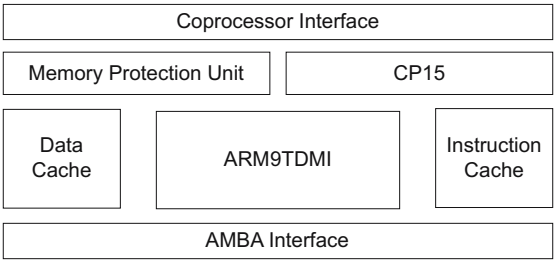


**Fig. 4.** Block diagram of the ARM940T architecture

The ARM9TDMI RISC-processor core consists of a five stage pipeline, which is controlled by a 32 bit instruction word. Each instruction word is derived from the standard ARM instruction set. The standard ARM instruction set itself is based on a load/store architecture. As a consequence, the source data of different instructions must be loaded separately into one or two source registers. The result is written back to a target register. Therefore, the instruction set can be divided into load/store and arithmetic instructions. However, branch and control instructions are supplied by the standard ARM instruction set. To improve the code density the ARM9TDMI processor core also features a dynamic instruction set exchange to the Thumb instruction set. These 16 bit instructions are compressed versions of a subset of the standard ARM instructions. The exchange is performed by dynamic decompression in the ARM9TDMI pipeline.

The CP15 coprocessor is accessed via the coprocessor interface using specific assembler directives. It enables and initializes the memory protection unit, which itself enables instruction and data regions in the main memory as well as in the instruction and data cache. Moreover, the memory protection unit sets and monitors access rules for the different instruction and data regions.

For the course of this modeling work, a so-called ARM Integrator Core Module featuring an ARM940T has been applied as reference platform. Internal processor states could be analyzed using a MultiICE in circuit emulation interface and an instruction set simulator (ARMulator).

## 5 Hybrid FLPA/ILPA Modeling of the ARM940T

In contrast to the FLPA modeling of some complex VLIW-DSP-architectures where the processor architecture had to be separated into up to seven functional blocks [5] for the modeling of the ARM940T only a separation into three different functional blocks is required. These are the ARM9TDMI processor core, the instruction cache and the data cache. According to the FLPA modeling concept each functional block is described by an arithmetic model, which itself describes the power consumption of the functional block. It can be found via simulations or measurements [5]. Hence, it is necessary to excite each block separately. This can be achieved by executing different parts of assembler code, which will be called scenarios in the following. Both, scenarios with and without cache misses

have to be considered to model the ARM9TDMI processor core (execution unit) and the instruction and data caches.

In Fig. 5 the power consumption is depicted as a function of the frequency of the core clock while the ARM940T is executing exemplary SUB and MUL scenarios without cache misses (test loops featuring 1000 SUBs (single cycle op.) or 1000 MULs (three cycle op.)). The results show that there are significant differences between individual instructions and that for each instruction a nearly perfect linear frequency dependency results (avg. coefficient of determination, a measure that is used to determine how well a regression fits [9], for all operation types is $R^2_{avg} = 0.9993$, theoretical maximum for $R^2$ is 1). In the following, the according power consumption of a test loop with an instruction $i$ at the frequency $f$ without cache misses is denoted as instruction-specific offset $P_{inst\_spec}(i, f)$.

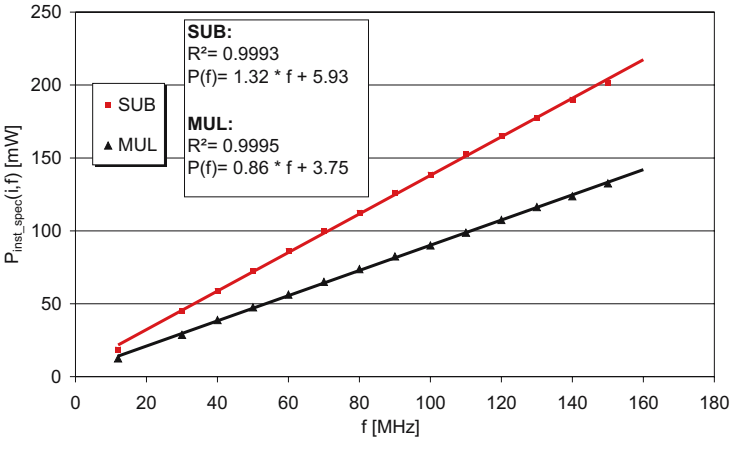


**Fig. 5.** Power consumption as a function of the frequency of the ARM940T core clock while executing SUB and MUL scenarios without cache misses

It has to be regarded, that the distribution of basic instructions significantly varies from application to application. Two exemplary distributions for a 4 tap 1D FIR filter and an MP3 (MPEG1/2 Layer3) decoder are depicted in Fig. 6.

Regarding the strong dependency of the power consumption on the operation which is performed and the significant difference of the distribution of instructions this shows, why it is required to extend the classical FLPA approach by an instruction dependent part. This new approach is denoted as hybrid FLPA/ILPA modeling. One key element of this approach is that for each application, respectively task, whose power consumption has to be determined, a dynamic determination of the distribution of instructions on the basis of the assembler code has to be performed.

Increasing the number of instructions in a test scenario (here more than 1024 instructions, due to the cache size and the instruction word length) leads to cache misses. As shown in Fig. 7, the power consumption is no longer a linear

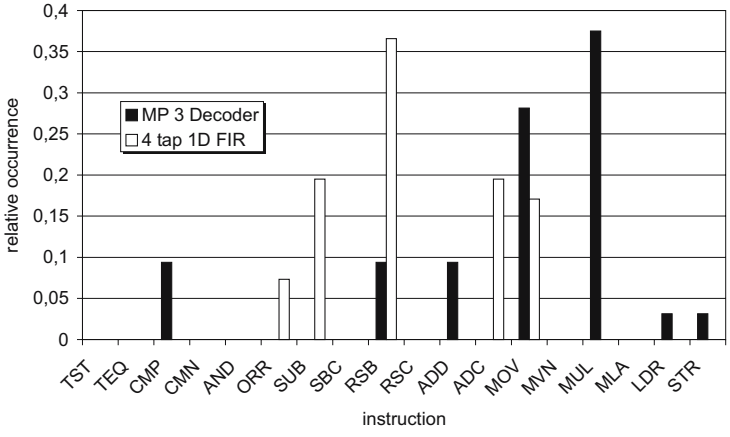


**Fig. 6.** The distribution of instructions within the most frequently used sections of the assembler code for a 4 tap 1D FIR filter and an MP3 decoder

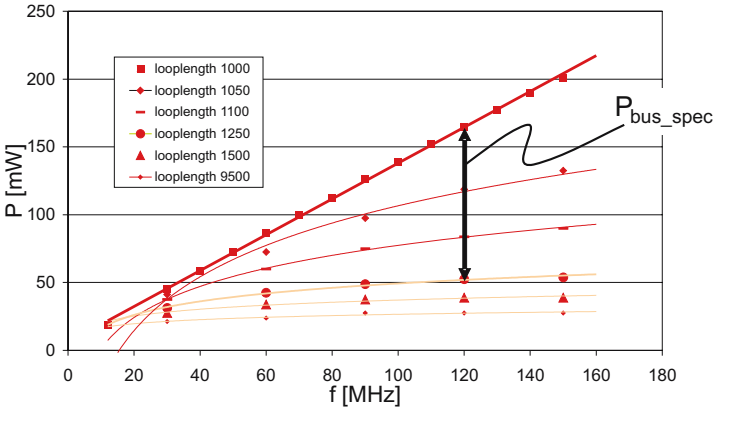


**Fig. 7.** Power consumption as a function of the frequency of the core clock while executing SUB test scenarios featuring cache misses

function of the core clock frequency while the processor is executing those test scenarios.

The difference at a given frequency between the instruction-specific offset and the actual power consumption of the ARM940T by executing such test scenarios with cache misses is called the bus-specific offset $P_{bus\_spec}$. Hence, the number of cache misses would be an appropriate parameter influencing the model for $P_{bus\_spec}$. Using the ARM instruction set simulator and cycle counter (ARMulator [10]) it is possible to derive various cycle counts (core clocks, memory bus clocks, etc.). These values are much more accurate than the number of cache misses which are also provided by the simulation environment [11]. Therefore,

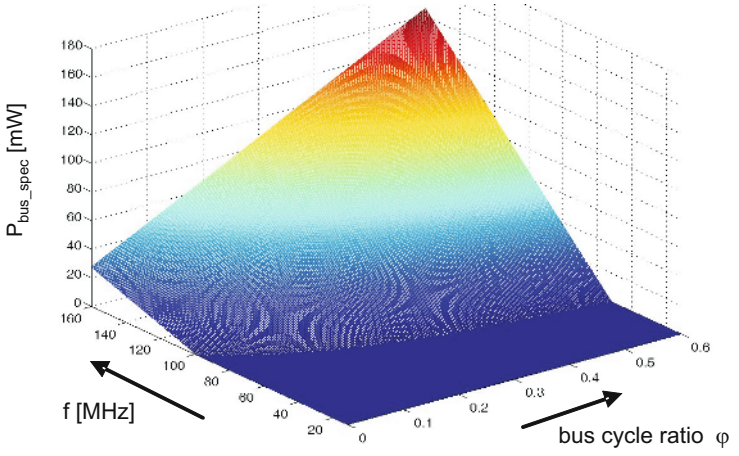


**Fig. 8.** Bus-specific offset of the ADD instruction

the bus-specific offset (see Fig. 8) can be modeled as a linear function of the ratio $S/T$. Here, the variable $S$ denotes the number of bus cycles that are followed by data movement and the variable $T$ denotes the total number of bus cycles. In the following, the ratio $S/T$ is denoted as the bus cycle ratio $\varphi$. Using the relative share of instruction cache misses $r_{icm}$ and the relative share of data cache misses $r_{dcm}$ the bus cycle ratio is split into a bus cycle ratio caused by instruction cache misses $\varphi_{inst}$ and caused by data cache misses $\varphi_{data}$, whereby the influence of the instruction and data caches apart from each other is considered. So with

$$\varphi_{inst} = r_{icm} \cdot \varphi \quad \text{and} \quad \varphi_{data} = r_{dcm} \cdot \varphi \tag{1}$$

the bus-specific offset is calculated by the equation

$$P_{bus\_spec} = P_{bus\_spec}(i, f, \varphi_{inst}) + P_{bus\_spec}(i, f, \varphi_{data}) \quad . \tag{2}$$

Besides the dependency on the bus cycle ratio $\varphi$ the bus-specific offset is also a function of the frequency. It could be modeled as

$$P_{bus\_spec}(f, \varphi) = a \cdot \varphi + b \cdot f + c \cdot \varphi \cdot f + d \quad . \tag{3}$$

Negative values for $P_{bus\_spec}(f, \varphi)$ are not possible and clipped to zero.
Finally, the actual power consumption of a given task can be calculated by

$$P_{act} = P_{inst\_spec} - P_{bus\_spec} \quad . \tag{4}$$

To estimate the complete power consumption of the ARM940T processor while executing a task, a profiler from the ADS1.2 framework [10] determines the share $h_{label}$ of the execution time of the different parts of the assembler code which are produced by the compiler and which are denoted here as labels. The instruction distribution is determined for every label by a special parser

**Table 1.** Subset of parameters of the ARM940T hybrid FLPA/ILPA model

| Instruction Class i | a | b | c | d |
|---|---|---|---|---|
| **arithmetic** (ADD, SUB, ...) | 43.07 | 0.451 | 1.49 | -44.6 |
| **logic** (AND, CMP, ...) | 48.68 | 0.462 | 1.42 | -47.3 |
| **multiplication** (MUL, MLA, ...) | 194.82 | 1.436 | -1.01 | -133.6 |
| **load** (LDR, LDRB, ...) | 82.54 | 0.61 | 1.61 | -60.7 |
| ... | ... | ... | ... | ... |

which has been implemented as a C program, whereby the complete share $h_i$ of every instruction class $i$ in the label is extracted. The parser categorizes the instruction set into 6 different instruction classes. It has been analyzed by comprehensive inspections that 6 different instruction classes is an attractive compromise between estimation accuracy and modeling effort. Using less instruction classes significantly decreases the estimation accuracy while increasing the number of instruction classes only very marginally improves the estimation accuracy. Some exemplary parameters which are required to calculate $P_{bus\_spec}(f, \varphi)$ for some exemplary instruction classes are depicted in Table 1. The resulting hybrid FLPA/ILPA power model of the ARM940T can be summarized as follows

$$P_{act}(i, f, \varphi) = \sum_{label} h_{label} \cdot (\sum_{i} h_i \cdot (P_{inst\_spec}(i, f) - P_{bus\_spec}(i, f, \varphi))) \qquad (5)$$

## 6 Benchmarking of the Hybrid FLPA/ILPA Model

The estimated power consumption was compared to the measured values for a variety of tasks in order to benchmark the hybrid model (see Fig. 9). The

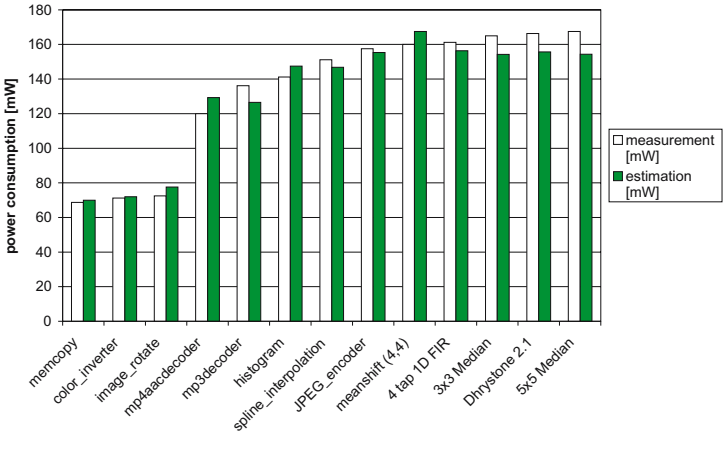


**Fig. 9.** FLPA estimation results and measurements for the AR940T architecture

comparison of estimated and measured values shows a maximum error of 7.8 % and an average error of 4.8 % for the power consumption.

As can be seen in Fig. 9, the variety of tasks which has been inspected on this platform features a dynamics concerning the according power consumption of more than 55 % (e.g. memcopy: 68.7 mW, 5x5 median: 165.0 mW). Thus, the estimation error is much smaller than the power consumption dynamics of the ARM940T. It is one of the key features of this modeling technique that it provides a very robust estimation accuracy while covering even a very wide range of applications with their according high power consumption dynamics. Some other available power models [12] can provide such an attractive estimation accuracy only for a limited range of applications with a corresponding small power dynamics. However, a power modeling approach can be applied only successfully if it is applicable also for a wide range of signal processing tasks.

## 7 Conclusion

Different approaches for power estimation for programmable processors have been analyzed. It has been shown that the concept of so-called Functional-Level Power Analysis (FLPA) has to be extended by an instruction dependent part in order to achieve high estimation accuracy even for embedded general purpose processor cores. According to this hybrid functional and instruction level modeling approach the processor architecture has been separated into several functional blocks. The power consumption of these blocks has been described in terms of parameterized arithmetic models. A parser which allows to analyze automatically the assembler codes has been implemented. This parser yields the input parameters of the arithmetic models like e.g. distribution of instructions or the type and number of memory accesses. An evaluation of this approach has been performed applying an ARM940T processor core and a variety of basic signal processing tasks. Resulting estimated power figures were compared to physically measured values. A maximum estimation error of 8 % for the absolute power consumption is achieved. This estimation error is much smaller than the dynamics of the power consumption for the inspected variety of tasks (55 %). The application of this methodology allows to evaluate efficiently different parameter settings of a programmable processor, different coding styles, compiler settings, algorithmic alternatives etc. concerning the resulting power consumption.

## References

1. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: A framework for architectural-level power analysis and optimizations. In: Proc. of the ISCA. (2000) 83–94
2. Tiwari, V., Malik, S., Wolfe, A.: Instruction level power analysis and optimization of software. Journal of VLSI Signal Processing **13** (1996) 1–18
3. Qu, G., Kawabe, N., Usami, K., Potkonjak, M.: Function level power estimation methodology for microprocessors. In: Proc. of the Design Automation Conference. (2000) 810–813

4. Senn, E., Julien, N., Laurent, J., Martin, E.: Power consumption estimation of a C program for data-intensive applications. In: Proc. of the PATMOS Conference. (2002) 332–341
5. Blume, H., Schneider, M., Noll, T.G.: Power estimation on a functional level for programmable processors. In: Proc. of the TI Devel. Conf., Houston. (2004)
6. von Livonius, J., Blume, H., Noll, T.G.: FLPA-based power modeling and power aware code optimization for a Trimedia DSP. In: Proc. of the ProRISC-Workshop, Veldhoven, Netherlands. (2005)
7. Furber, S.: ARM System-on-Chip Architecture. Addison-Wesley (2000)
8. ARM: ARM940T Tech. Ref. Manual, Rev2, ARM DDI 0144B. (2000)
9. Sachs, L.: Angewandte Statistik (in German). Springer Verlag (1996)
10. ARM: RealView ARMulator ISS User Guide, V. 1.4, ARM DUI 0207C. (2004)
11. ARM: App. Note 93 Benchmarking with ARMulator, ARM DAI 0093A. (2002)
12. Senn, E., Julien, N., Laurent, J., Martin, E.: Functional level power analysis: An efficient approach for modeling the power consumption of complex processors. In: Proc. of the IEEE DATE. (2004) 666–667