Journal of Embedded Computing 1 (2005) 317–324
IOS Press

Instruction level energy modeling for pipelined processors[1]

S. Nikolaidis[a,*], N. Kavvadias[a], T. Laopoulos[a], L. Bisdounis[b] and S. Blionas[b]
[a]*Department of Physics, Aritstotle University of Thessaloniki, 54124 Thessaloniki, Greece*
[b]*INTRACOM S.A., Peania, Greece*

Abstract. A new method for creating instruction level energy models for pipelined processors is introduced. This method is based on measuring the instantaneous current drawn by the processor during the execution of the instructions. An appropriate instrumentation set up was established for this purpose. According to the proposed method the energy costs (base and inter-instruction costs) are modeled in relation to a reference instruction (e.g. NOP). These costs incorporate inter-cycle energy components, which cancel each other when they are summed to produce the energy consumption of a program resulting in estimates with high accuracy. This is confirmed by the results. Also the dependencies of the energy consumption on the instruction parameters (e.g. operands, addresses) are studied and modeled in an efficient way.

Keywords: Instruction-level energy modeling, energy characterization, instantaneous current measurement, microprocessor

1. Introduction

EMBEDDED computer systems are characterized by the presence of dedicated processors which execute application specific software. A large number of embedded computing applications are power or energy critical, that is power constraints form an important part of the design specification [1]. Recently, significant research in low power design and power estimation and analysis has been developed. The determination of methods for the accurate calculation of the energy consumption in processors is required for system level energy consumption evaluation and optimization.

Power analysis techniques for embedded processors that employ physical measurements were firstly suggested in mid 90's. Significant effort on software optimization for minimizing power dissipation is found in [1–4] where a technique based on physical measurements is developed. Power characterization is done with the extraction of cost factors for the average current drawn by the processor as it repeatedly executes short instruction sequences. The base cost for an instruction is determined by constructing a loop with several instances of the same instruction. Inter-instruction effect induced when executing different adjacent instructions, is measured by replacing the one-instruction loops used in the measurement of base costs, with loops consisting of appropriate instruction pairs. The sum of the power base cost of the instructions executed in a program, refined by the power cost of the inter-instruction effects, are considered to provide the power cost of the program. This method has been validated for commercial targets based on embedded core processors.

The majority of work published on the field of measurement-based techniques, refers to the Tiwari method [1] as a base point. By Tiwari method only average power estimates can be utilized for modeling task, since the measurements are taken with a standard digital ammeter. Direct application of the Tiwari technique is found in [5] where an extensive study of the ARM7TDMI processor core is reported. In [6], physical measurements for the processor current are also obtained by a precise amperemeter. However, power modeling effort is more sophisticated, as architectural-

[1]This work was supported by EASY project, IST-2000-30093, funded by the European Union.
*Corresponding author. Tel.: +30 2310 998078; Fax: +30 2310 998018; E-mail: snikolaid@physics.auth.gr.

ISSN 1740-4460/05/$17.00

level model parameters are introduced and integrated within the power model. These consist of the weight of instruction fields or data words, the Hamming-distance between adjacent ones, and basic costs for accessing the CPU, external memory and activating/deactivating functional units.

Instantaneous current is firstly measured in [7], where a digitizing oscilloscope is used for reading the voltage difference over a precision resistor that is inserted between the power supply and the core supply pin of the processor. Instantaneous power is then calculated directly from the voltage waveform from which average figures are extracted to guide instruction power modeling. Resistor-based methodologies suffer from supply voltage fluctuations over the processor which reduces the accuracy of the method.

All the above techniques acquire the current drawn by the processor on instruction execution. A complex circuit topology for cycle-accurate energy measurement is proposed in [8,9], which is based on instrumenting charge transfer using switched capacitors. The switches repeat on/off actions alternately. A switch pair is charged with the power supply voltage during a clock cycle and is discharged during the next cycle powering the processor. The change in the voltage level across the capacitors is used for the calculation of the consumed energy in a clock cycle. However, this method can not provide detail information for the shape of the current waveform, which may be significantly useful in some applications and also in case high quality power models including the architectural characteristics of the processor are required. In order to measure the energy variations, various (*ref*, *test*) instruction pairs are formed, where *ref* notes a reference instruction of choice and test the instruction to be characterized. This setup combined with the above modeling concept are then utilized to obtain an energy consumption model for the ARM7 processor.

In this paper, a new instruction-level energy modeling methodology for simple in-order pipelined processors is proposed. The derived energy models are used for software energy consumption estimation. The proposed methodology is based on measurements of the instantaneous current of the processor. A simple measuring environment was established for this purpose [10,11]. A high-speed current mirror is used as current sensing circuit, eliminating the voltage fluctuation problem mentioned for the technique in [7]. The current waveform is integrated in a clock period and the energy consumption of the processor in a clock cycle is calculated. Based on these calculations and executing appropriate instruction combinations the energy costs of any instruction are defined and modeled. These costs correspond to base costs, inter-instruction costs and costs due to idle cycles of the processor. Also, the base costs are refined with the contribution of the effect of the instruction parameter values on the instruction energy. Only the number of 1's in the word space of these parameters is required for determining their energy consumption.

This paper is organized as follows. In Section 2 the modeling methodology is presented. The measured energy consumed during the execution of specific instructions is analyzed in its components leading to the definition of the energy costs of the instructions. In Section 3 the expression of the energy of an instruction as well as of a software program including all the relative factors are provided. The ARM7TDMI embedded processor is used as a case study and results for the base and inter-instruction costs and an analysis of them are presented in Section 4. Also, the effect of the pipeline stalls on the energy consumption is characterized. In Section 5 the influence of the instruction parameters on the energy consumption is determined and modeled. Finally, we conclude in Section 6.

2. Instruction-level energy modeling for pipelined processors

The proposed method is based on the measurement of the instantaneous current drawn by the processor during the execution of the instructions. The instrumentation set-up for measuring the instantaneous current has been presented in [10,11]. The current sensing circuit is a high performance current mirror which copies the drawn current on a resistor. A high-speed digitized oscilloscope is used for monitoring the voltage drop on the resistor. Taking the voltage waveform and making the appropriate process calculations the energy consumption at each clock cycle can be derived.

The method is developed for pipelined processors like the ARM7 (three-stage pipeline). However, its application for non-pipelined processors is straightforward.

2.1. Energy consumed in pipelined processors

The current drawn by the processor in a clock cycle is the result of the concurrent process of many instructions due to the pipeline structure of the processor. So, it is rather impossible to isolate the current drawn from

Pipeline Stages	**3-stage pipeline operation**				
IF	**NOP**	**NOP**	**Instr**	**NOP**	**NOP**
ID	**NOP**	**NOP**	**NOP**	**Instr**	**NOP**
EX	**NOP**	**NOP**	**NOP**	**NOP**	**Instr**
Clock Cycles	**n-1**	**n**	**n+1**	**n+2**	**n+3**

Fig. 1. Pipeline states during the execution of instructions.

the pipeline stage, which processes the test instruction. Instead, we have to find a method for calculating the energy consumed by an instruction from the current drawn in the clock cycles needed for its execution.

In each clock cycle the instantaneous current, $i(t)$ is monitored and the energy consumed in that cycle, E_{cycle}, is calculated as:

$$E_{cycle} = V_{DD} \int_0^T i(t)dt \tag{1}$$

where V_{DD} is the value of the supply voltage and T is the clock period. The energy which is consumed as an instruction passes through a n-stage pipeline, $E\_M(Instr)$, is distributed in n clock cycles and it is found by summing the energy components of the n cycles:

$$E\_M(Instr) = E_{cycle\_1} + E_{cycle\_2} + \ldots + E_{cycle\_n}. \tag{2}$$

In n clock cycles n instructions are executed. Suppose a set of instructions with only one instance of the *test* instruction, and with the others being the same, let us say a reference instruction (probably the NOP), running through the pipeline. Then in n clock cycles one *test* instruction and $n-1$ reference instructions are executed, and the energy corresponding to the test instruction is calculated as:

$$E(Instr) = E\_M(Instr) - (n-1)E(ref). \tag{3}$$

Since the energy budget of a program corresponds to the sum of the energy of each instruction, such a model seems to lead to an overestimation because instead of modeling the circuit state changes from one instruction to another, we model every instruction as a circuit state change (change from test to NOP instruction), which results in counting the number of circuit state changes twice. To overcome this problem the inter-instruction effect is also modeled in a similar way, and in this case the value of this effect is expected to be rather (at least in most cases) negative compensating this overhead.

2.2. Base instruction energy cost modeling

In the proposed method as base instruction cost we mean the amount of energy which is consumed for the execution of an instruction after the execution of a reference instruction (NOP). In order to calculate the base instruction costs, loops including an instance of the test instruction and a number of instances of reference instruction are executed on the processor. Figure 1 shows the pipeline states during the clock cycles required for the execution of the instruction on a three-stage pipelined processor. The instruction is executed in three cycles. In each clock cycle the current is monitored and the energy is calculated.

The energy consumed in these three stages is calculated as:

$$E\_M(Instr) = E_{cycle\_n+1} + E_{cycle\_n+2} + E_{cycle\_n+3}. \tag{4}$$

In these three cycles one test instruction and two reference instructions are executed. This amount of energy contains inter-cycle effects due to the change of the pipeline states and it can be expressed as

$$E\_M(Instr) = E_{Instr} + 2E_{NOP} + E_{NOP,Instr} + E_{Instr,NOP} + E_{NOP,NOP} \tag{5}$$

where E_{inst} corresponds to the real energy of the instruction, E_{NOP} to the energy of the NOP instruction and the others to the inter-cycle effects between the cycles $n, n+1 (E_{NOP,Instr})$, $n+1, n+2 (E_{Instr,NOP})$ and $n+2, n+3 (E_{NOP,NOP})$.

The energy of the instruction is calculated as:

$$E(Instr) = E\_M(Instr) - 2E_{NOP} \tag{6}$$

and corresponds to

$$E(Instr) = E_{Instr} + E_{NOP,Instr} + E_{Instr,NOP} + E_{NOP,NOP}. \tag{7}$$

The energy of the NOP instruction is:

$$E_{NOP} = E_{cycle\_n}. \tag{8}$$

2.3. Inter-instruction effect cost modeling

The instruction sequence for measuring the inter-instruction effect is shown in Fig. 2. Appropriate loops will be executed on the processor. In this case, in four clock cycles one *Instr1*, one *Instr2* and two reference instructions are executed.

Pipeline Stages	**3-stage pipeline operation**				
IF	**NOP**	**Instr1**	**Instr2**	**NOP**	**NOP**
ID	**NOP**	**NOP**	**Instr1**	**Instr2**	**NOP**
EX	**NOP**	**NOP**	**NOP**	**Instr1**	**Instr2**
Clock Cycles	**n**	**n+1**	**n+2**	**n+3**	**n+4**

Fig. 2. Pipeline states during the execution of instructions for measuring inter-instruction costs.

The instruction sequence for measuring the inter-instruction effect is shown in Fig. 2. Appropriate loops will be executed on the processor. In this case, in four clock cycles one *Instr1*, one *Instr2* and two reference instructions are executed.

Consequently, it is:

$$E\_M(Instr1, Instr2) = E_{cycle\_n+1} + E_{cycle\_n+2} + E_{cycle\_n+3} + E_{cycle\_n+4} \quad (9)$$

which corresponds to:

$$E\_M(Instr1, Instr2) = E_{Instr1} + E_{Instr2} + 2E_{NOP} + E_{NOP,Instr1} + E_{Instr1,Instr2} + E_{Instr2,NOP} + E_{NOP,NOP} \quad (10)$$

while the inter-instruction effect is figured as:

$$E(Instr1, Instr2) = E\_M(Instr1, Instr2) - E(Instr1) - E(Instr2) - 2E_{NOP} \quad (11)$$

and corresponds to:

$$E(Instr1, Instr2) = E_{Instr1,Instr2} - E_{Instr1,NOP} - E_{NOP,Instr2} - E_{NOP,NOP} \quad (12)$$

2.4. Modeling the energy of a program

The total energy consumed when a program is executed can be estimated by summing the base costs of the instructions and the inter-instruction costs. Doing that, it is observed that the inter-cycle costs included in the expressions Eqs (7) and (12) cancel each other and finally a sum including only the actual energy components of each instruction and the corresponding inter-instruction effects appears. Indeed, only three additional components appear, one inter-cycle component of the first instruction of the program and one of the last instruction and an inter-instruction effect between NOP instructions. These three components will determine the maximum theoretic error of the method. An example is given in Fig. 3 where the application of the method for a program consisting of three instructions ADD, MOV and CMP is analyzed. As it is shown, all the inter-cycle costs included in Eqs (7) and (12) are eliminated except the three ones mentioned before. In this way the actual energy components of the instructions are isolated from secondary effects and high accurate estimates for the processor can be created.

It has to be mentioned that a hypothesis was used so that the inter-cycle costs can cancel each other. The inter-cycle effect as shown in the expressions of the base cost and the inter-instruction cost is not the same since in these two cases the pipeline stages do not necessary correspond to the same state. However, according to our results this approximation does not influence significantly the accuracy of the method.

3. Accurate instruction-level energy modeling

As it was mentioned above, the energy consumed during the execution of instructions can be distinguished in two amounts. The base cost, energy amount needed for the execution of the operations which are imposed by the instructions, and the inter-instruction cost which corresponds to an energy overhead due to the changes in the state of the processor provoked by the successive execution of different instructions. Measurements for determining these two energy amounts for each instruction of the ARM7TDMI processor were taken and presented in [12]. However the base costs in [12] were for specific operand and address values (zero operand and immediate values and specific address values to minimize the effect of 1s). This base cost is called *pure* base cost.

We have observed in our measurements that there is a dependency of the energy consumption of the instructions on the values of their parameters (operand values, addresses). To create accurate models this dependency has to be determined. Consequently the dependency of the energy of the instructions on the register numbers, register values, immediate values, operand values, operand addresses and fetch addresses was studied. Additional measurements were taken to satisfy this necessity. It was observed by the measurements that there is a close to linear dependency of the energy on these parameters, versus the number of 1s in their word space. In this way the effect of any of the above *energy-sensitive factors* was efficiently modeled by a coefficient. Incorporating these effects in our models the proposed method keeps its promised accuracy while it becomes very attractive since it can be easily implemented in software as an estimation tool.

Table 1
Comparison of measured to calculated instruction energy costs (njoules) due to additional dependencies

Instruction formation	Measured		Calculated		% diff
	$E_{base}+E_{oprd}$	$E_{base}+E_{oprd}+E_{regn}$	$E_{base}+E_{oprd}$	$E_{base}+E_{oprd}+E_{regn}$	
ADD Rd,Rn,Rs,ASR Rm	2.58	2.61	2.57	2.61	−0.04
ADDRd,Rn,Rs,ASR imm	1.57	1.60	1.55	1.60	−0.27
ADD Rd,Rn,Rs	1.51	1.59	1.51	1.56	1.71
ADD Rd,Rn,Rs,RRX	1.51	1.63	1.52	1.64	−1.19
LDR Rd, [Rn,Rs]	3.07	3.27	2.97	3.29	−0.63
STR Rd, [Rn,Rs]	2.28	2.48	2.23	2.43	2.01

$$
\begin{aligned}
&E(ADD) = E_{ADD} + E_{NOP,ADD} + E_{ADD,NOP} + E_{NOP,NOP}\\
&E(MOV) = E_{MOV} + E_{NOP,MOV} + E_{MOV,NOP} + E_{NOP,NOP}\\
&E(CMP) = E_{CMP} + E_{NOP,CMP} + E_{CMP,NOP} + E_{NOP,NOP}\\
&E(ADD,MOV) = E_{ADD,MOV} - E_{ADD,NOP} - E_{NOP,MOV} - E_{NOP,NOP}\\
+\quad &E(MOV,CMP) = E_{MOV,CMP} - E_{MOV,NOP} - E_{NOP,CMP} - E_{NOP,NOP}\\
\hline
&E(\mathrm{Pr}ogram) = E_{ADD} + E_{MOV} + E_{CMP} + E_{ADD,MOV} + E_{MOV,CMP}\\
&\qquad + E_{NOP,ADD} + E_{CMP,NOP} + E_{NOP,NOP}
\end{aligned}
$$

Fig. 3. Estimation of the energy of a program consisting of three instructions.

Making some appropriate experiments we observed that the effect of each energy-sensitive factor on the energy cost of the instruction is independent of the effect of the other factors. The effects on the energy of these factors are uncorrelated as can be observed in Table 1. The distortion of our results from this conclusion is, most of the time, less than 2–3% and only in some marginal cases becomes more than 7%. According to this conclusion, the effect of the energy-sensitive factors can simply be added to give the total energy amount.

Other sources of energy consumption are conditions of the processor which lead to an overhead in clock cycles because of the appearance of idle cycles. This is the case of the appearance of pipeline stalls, which was measured and modeled.

According to the above, the energy, E_i, consumed during the execution of the i instruction can be modeled as:

$$E_i = b_i + \sum_j a_{i,j} N_{i,j} \tag{13}$$

where b_i is the pure base cost of the i instruction, $a_{i,j}$ and $N_{i,j}$ is the coefficient and the number of 1s of the j energy-sensitive factor of the i instruction, respectively.

Each instruction may contain some of these energy-sensitive factors. The effect of all the factors included in an instruction have to be taken into account to create the energy budget of the instruction.

Having modeled the energy cost of the instructions, the energy consumed for running a program of n instructions can be estimated as:

$$PE = \sum_1^n E_i + \sum_1^{n-1} O_{i,i+1} + \sum \varepsilon \tag{14}$$

where $O_{i,j}$ is the inter-instruction cost of the instructions i and j, and ε is the cost of a pipeline stall.

4. Pure base cost and inter instruction cost models – results

The complete models for the instruction-level energy consumption of the ARM7TDMI created according to the proposed methodology can be found in [13]. Pure base costs of all the instructions and for all the addressing modes are given. Since the number of the possible instruction pairs (taking into account the addressing modes) is enormous, groups of instructions and groups of addressing modes according to the resources they utilize, have been formed and inter-instruction costs have been given only for representatives from these groups. In this way we keep the size of the required model values reasonable without significant degradation of the

accuracy (error less than 5% in the inter-instruction cost by using representative instructions).

For the instructions which are executed in the same number of cycles the values of the pure base costs present in most cases a distortion less than 20% of a mean value. Also, it has been shown that the existence of a condition does not influence significantly the energy of the instruction. For the instructions which require more cycles (clock per instruction more than 1, CPI$>$1) the energy cost increases according to the required cycles. In this way, a computationally efficient model with less but not unacceptable accuracy (depending on the aim of its use) can be provided by assigning energy values to the instructions only according to the number of cycles they need.

Many values of the inter-instruction costs have negative sign as it was expected. The contribution of the inter-instruction costs, as they are calculated according to the proposed method, remains small. As it can be observed by our models most of the inter-instruction costs are less than 5% of the corresponding pure base costs while almost all the cases are covered by a 15% percentage. Also, there is no symmetry in the inter-instruction cost for a pair of instructions. For example, the execution of the LDR after the ADD present a cost of 0.064nJoule while the execution of ADD after LDR present a cost of -0.122nJoule. This seems more reasonable from the case of symmetric inter-instruction costs as Tiwari method supposes.

The ARM processor is a RISC 3-stages pipelined processor. Generally, the processing flow can be delayed due to a condition that cannot be satisfied at the time needed. In the case of the ARM processor core, we distinguish two categories of such conditions, a) pipeline flushes due to altering the control flow of the program by introducing a non-sequential value to the PC and b) pipeline stalls when the condition defined for a conditionally executed instruction is not satisfied.

For these types of pipeline stalls, small program loops to activate corresponding conditions have been implemented in order to measure their effect on the energy consumption. For each type of these stall cases in the pipeline, it was found that a single absolute overhead to describe their cost in energy is sufficient. The corresponding overhead values are shown in Table 2.

To determine the accuracy of the method a number of programs with various instructions have been created. In these instructions the operand values were kept zero and thus the effect of energy sensitive factors wasn't taken into account. The energy consumed during the execution of each program was calculated directly from measurements of the instantaneous drawn current and also calculated by using the derived instruction-level energy models. The error was found to be up to 1.5%.

Table 2
Model parameters for the effect of pipeline stalls

Instruction type	Absolute overhead to base cost (nJoules)
Pipeline flush (any)	2.04
Pipeline stalls	
Data processing and load-store	1.08
Multiply	1.46
Branch	1.60

5. Models of the energy sensitive factors effect – results

The dependency on the energy sensitive factors was also studied. Energy depends on the number of 1s in the word structures of these entities. The Hamming distance between the corresponding word fields of successive instructions is not considered here since according to the followed modeling methodology, where NOP is used as a reference instruction, Hamming distance equals to the number of 1s of the corresponding fields of the instructions.

The effect of each factor was studied separately from the others since, as it can be verified by the results, the correlation among the effect of these factors is insignificant. This energy dependency can be approximated with sufficient accuracy by linear functions. Coefficients were derived for each instruction for any energy sensitive factor. However, appropriate grouping of the instructions is used to keep reasonable the number of required coefficients to increase the applicability of the method without significant loss in the accuracy.

In Fig. 4 the effect of the register number for data-processing instructions in register addressing mode is presented. The error in estimation for the selected coefficient is less than 3%. Such a value of the error characterizes all the selected coefficients. The energy consumption versus the number of 1s in operand values for the STR instruction is shown in Fig. 5. The linear dependency is obvious in both figures. Such a linearity has been observed and for the rest factors.

To evaluate the absolute accuracy of our modeling approach, real kernels were used as benchmarks. The corresponding assembly list has been extracted from C programs by utilizing the facilities of the *armcc* tool, shipped with the ARM ADS software distribution. The energy consumption during the cycles of the program

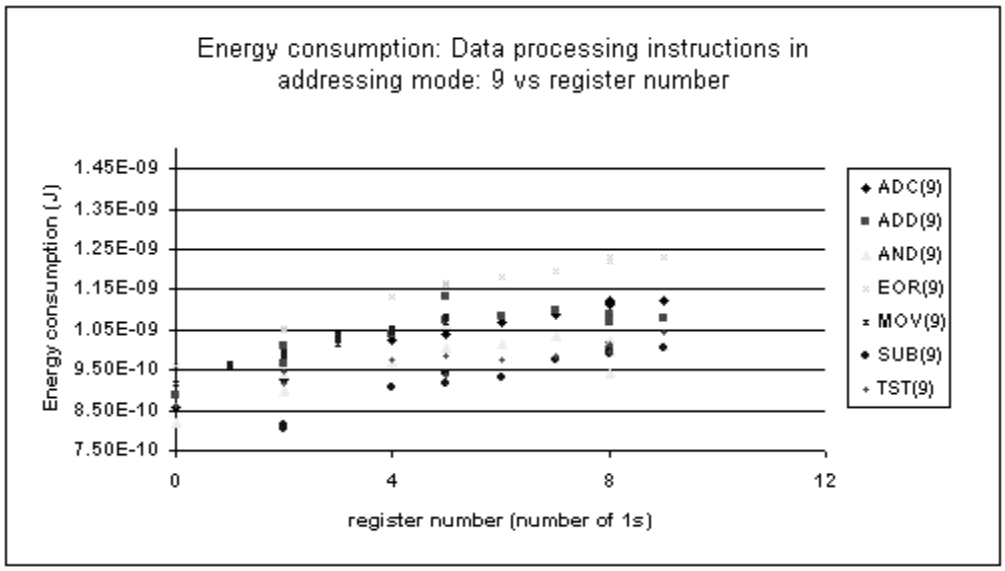

Fig. 4. The effect of register number for data-processing instructions in register addressing mode.

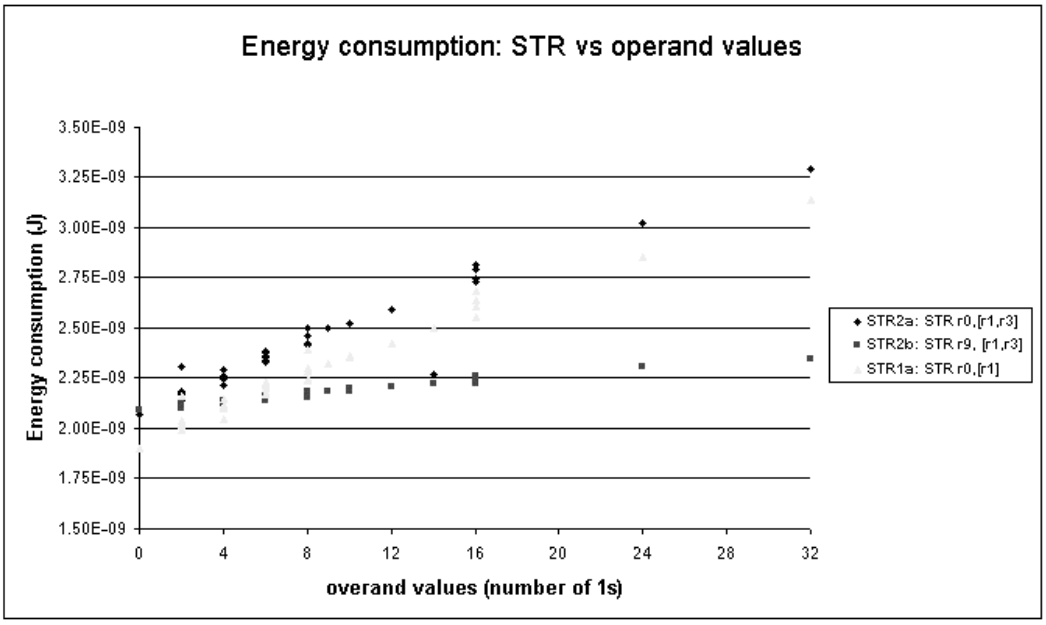

Fig. 5. Energy consumption versus the number of 1s in operand values for the STR instruction.

was calculated by direct measurement of the drawn current and by using the derived models and Eq. (14). In Table 3 the measured and estimated energy consumption for five common software kernels are presented. According to our results the error of our modeling approach in real programs was found to be less than 5–6%.

6. Conclusion

High quality instruction level energy models can be derived for pipelined processors by monitoring the instantaneous current drawn by the processor at each clock cycle. Knowing the current waveform the corresponding energy consumed at a clock cycle can be

Table 3
Model papameters for the effect of pipeline stalls

	Program energy consumption		
Benchmark	Estimated (nJ)	Measured (nJ)	error %
Cadd	32.78	33.43	1.94
Cmul	13.17	12.73	−3.46
Fir	46.51	44.70	−4.05
Sad	52.04	52.01	−0.06
Ablend	22.35	23.76	5.93

calculated. However, energy components of an instruction exist in different clock cycles. An efficient method is proposed to extract the actual energy components, which are used for the calculation of the energy consumption of complete software programs. The instruction base costs and inter-instruction costs are modeled in relation to a reference instruction. The secondary effects existing in these models (inter-cycle effects due to pipeline) eliminate each other when they are summed to give the total energy of programs, minimizing the error of the estimation. This method has been applied to the ARM7TDMI processor. Also, the dependency of the instruction energy on the instruction parameters (e.g. register numbers, register values, operand values) was studied. This dependency is approximated with linear functions and the appropriate coefficients are extracted. Finally, the application of the method in real kernels shows up to 5–6% error in the estimation of the energy consumption.

References

[1] V. Tiwari, S. Malik and A. Wolfe, Power analysis of embedded software: A first step towards software power minimization, *IEEE Trans. on Very Large Scale Integration* (*VLSI*) *Systems* **2**(4) (Dec. 1994), 437–445.

[2] V. Tiwari, S. Malik, A. Wolfe and M.T.-C. Lee, Instruction level power analysis and optimization of software, *Journal of VLSI Signal Processing* **13**(2–3) (Aug. 1996), 223–238.

[3] M.T.-C. Lee, V. Tiwari, S. Malik and M. Fujita, Power analysis and minimization techniques for embedded DSP software, *IEEE Trans. on Very Large Scale Integration* (*VLSI*) *Systems* (Mar. 1997), 123–135.

[4] V. Tiwari and T.C. Lee, Power analysis of a 32-bit embedded microcontroller, *VLSI Design Journal* **7**(3) (1998).

[5] SOFLOPO, *Low Power Development for Embedded Applications*, ESPRIT project: Deliverable 2.2: Physical measurements. By Thanos Stouraitis, University of Patras, Dec. 1998.

[6] S. Steinke, M. Knauer, L. Wehmeyer and P. Marwedel, *An accurate and fine grain instruction-level energy model supporting software optimizations*, Int. Workshop on Power and Timing Modeling, Optimization and Simulation, Yverdon-les-bains, Switzerland, Sep. 2001.

[7] J.T. Russell and M.F. Jacome, *Software power estimation and optimization for high performance, 32-bit embedded processors*, Int. Conference on Computer Design, Oct. 1998.

[8] N. Chang, K. Kim and H.-G. Lee, Cycle-accurate energy consumption measurement and analysis: Case study of ARM7TDMI, *IEEE Trans. on VLSI Systems* **10**(2) (Apr. 2002), 146–154.

[9] S. Lee, A. Ermedahl, S.-L. Min and N. Chang, *An accurate instruction-level energy consumption model for embedded RISC processors*, ACM SIGPLAN Workshop on Languages, Compilers and Tools for Embedded Systems, 2001.

[10] T. Laopoulos, P. Neofotistos, K. Kosmatopoulos and S. Nikolaidis, Measurement of current variations for the estimation of software-related power consumption, *IEEE Trans. on Instrumentation and Measurement* **52**(4) (Aug. 2003), 1206–1212.

[11] S. Nikolaidis, N. Kavvadias, P. Neofotistos, K. Kosmatopoulos, T. Laopoulos and L. Bisdounis, *Instrumentation set-up for instruction level power modeling*, Int. Workshop on Power and Timing Modeling, Optimization and Simulation, Seville, Spain, Sep. 2002.

[12] S. Nikolaidis, N. Kavvadias and P. Neofotistos, *Instruction level power measurements and analysis*, IST-2000-30093/EASY Project, Deliverable 15, Sep. 2002. http://easy.intranet.gr.

[13] S. Nikolaidis, N. Kavvadias and P. Neofotistos, *Instruction level power models for embedded processors*, IST-2000-30093/EASY Project, Deliv. 21, Dec. 2002. http://easy.intranet.gr.