Measurements Analysis of the Software-Related Power Consumption in Microprocessors

N. Kavvadias, P. Neofotistos, S. Nikolaidis, K. Kosmatopoulos and Th. Laopoulos

Electronics Lab. Physics Dept., Aristotle University of Thessaloniki,

Thessaloniki, 54124, Greece,

Email: laopoulos@physics.auth.gr

ABSTRACT

In this paper the measurements taken for the development of instruction-level energy models for microprocessors are presented and analyzed. An appropriate measuring environment and a suitable measuring methodology was developed for taking the necessary measurements. The energy of an instruction is defined as a sum of three components which are the pure base energy cost, the inter-instruction cost and the effect of the energy sensitive factors (instruction parameters). These components are characterized for each instruction of the ARM7TDMI embedded processor and their values are analyzed. Using the resulted models estimates of the energy consumption of real software kernels with only up to 6% error was determined.

1.INTRODUCTION

Last years, low power consumption has been established as the third main design target for digital systems together with performance and area. Consequently, accurate estimators of the power consumption at the system as well as the lower levels of abstraction are necessary. A large number of embedded computing applications are power or energy critical. Early work on processor analysis had focused on performance improvement without determining the power-performance tradeoffs. Recently, significant research in low power design and power estimation and analysis has been developed.

Accurate models for the energy consumed during the execution of software by the processor, termed as software energy consumption, are needed for the design of optimized, in terms of power, software structures. Since software energy contributes significantly to the total system energy the necessity for such accurate models increases.

Embedded software energy modeling techniques are distinguished into two main categories, physical measurement-based and simulation-based ones. In measurement-based approaches [1-7], the energy consumption of software is characterized by data obtained from real hardware. The advantage of measurement-based approaches is that the resulting energy model proves close to the actual energy behavior of the processor. In simulation-based methods, energy consumed by software is estimated by calculating the energy consumption of various components in the target processor through simulations at different levels [8-10]. However, the circuit-level implementation data for off-the-self microprocessors are not generally available, which makes the simulation-based approach more difficult to apply.

In measurement techniques, a common practice is to associate instructions running on the processor with their corresponding energy cost. The majority of work published on the field of measurement-based techniques, refers to the Tiwari method [1] as a base point. By this method only average power estimates can be utilized for modeling tasks, since the measurements are taken with a standard digital ammeter. Loops of hundreds of instances of the same instruction are performed on the processor and the average drawn current is used to model the energy consumption of the instruction. However, such experiments do not correspond to real processor execution conditions.

Direct application of the Tiwari technique is found in [11,12] where an extensive study of the ARM7TDMI processor core is reported. In [5], physical measurements for the processor current are also obtained by a precise amperemeter. However, power modeling effort is more sophisticated, as architectural-level model parameters are introduced and integrated within the power model. Instantaneous current is firstly measured in [6], 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 pins 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 voltage fluctuations over the supply voltage of the processor, which reduces the accuracy of the method [13].

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 [7], which is based on instrumenting charge transfer using switched capacitors. The switches repeat on/off actions alternately. A switched capacitor 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 proportional to the square of the consumed energy and this value is used for the calculation of energy in a clock cycle. 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. The above setup is utilized to obtain an energy consumption model for the ARM7 processor. However, this method employs a complex measuring instrumentation and also cannot provide detailed information for the shape of the current waveform, which may be useful in some cases.

Accurate instruction-level power or energy models can be created by accurately monitoring the instantaneous current drawn by the processor and then calculating the energy of the instruction. For this purpose an appropriate, simple and accurate measuring environment has been proposed by the authors in [13,14] using a high performance current mirror based on BJT transistors as current sensing circuit, eliminating the voltage fluctuation problem of previously mentioned techniques. The current waveform is integrated in a clock period and the energy consumption of the processor in a clock cycle is calculated. An instruction-level energy consumption modeling methodology has been proposed [15] aiming in the creation of highly accurate models. This methodology has been developed for in-order pipelined processors, like ARM7TDMI, which often is met in embedded applications. According to this methodology the instructions are characterized with energy values taken by applying realistic conditions on the processor. The energy consumed during the execution of an instruction is analyzed in its components. Appropriate experiments are performed and these components are modeled. Also, the contribution of the effect of the instruction parameter values is determined and modeled in an efficient way. In this paper the results of our experiments are presented and analyzed.

2. INSTRUCTION-LEVEL ENERGY MODELING

The energy consumed during the execution of instructions can be distinguished in two components. 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 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 (0.35um process) were taken and presented in [16]. However, the base costs were for specific operand and address values (zero operand and immediate values and specific address values to minimize the number of 1s in their word space). This base cost is called *pure base cost*.

It has been observed in measurements that there is a dependency of the energy consumption of the instructions on the values of their parameters (register number, operand values, operand addresses, etc). This dependency has to be determined in order to create accurate models. Additional measurements were taken to satisfy this necessity. It was noticed that there is a close to linear dependency of the energy on these parameters, versus the number of 1s in their word space. Consequently, the effect of any of the above *energy-sensitive factors* can be efficiently modeled by a coefficient.

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 shown in Table 1. (As *opval*, we denote the energy-sensitive factor corresponding to operand values, whereas *regnum* corresponds to the register number). The distortion of the 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, for example, is the case of the appearance of pipeline stalls. The effect of such cases on the energy consumption was measured and modeled.

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

	Measured		Calculated		
Instruction formation	opval	opval+ regnum	opval	opval+ regnum	% diff
ADD R_d,R_n,R_s,ASR R_m	2.58	2.61	2.57	2.61	-0.04
ADD R_d,R_n,R_s, ASR #imm	1.57	1.60	1.55	1.60	-0.27
ADD R_d,R_n,R_s	1.51	1.59	1.51	1.56	1.71
ADD R_d,R_n,R_s,RRX	1.51	1.63	1.52	1.64	-1.19
LDR R_d, [R_n,R_s]	3.07	3.27	2.97	3.29	-0.63
STR R_d, [R_n,R_s]	2.28	2.48	2.23	2.43	2.01

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

$$E_i = b_i + \sum_j a_{i,j} N_{i,j} \tag{1}$$

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. The coefficients for the energy-sensitive factors are given in Table 2. Each instruction may contain some of these energy-sensitive factors, which have to be taken into account to create the energy budget of the instruction.

Table 2. Energy-sensitive factor coefficients

Energy-sensitive factor	Coefficient
Register number	$a_{i,1}$
Register value	$a_{i,2}$
Immediate value	$a_{i,3}$
Operand value	$a_{i,4}$
Operand address	$a_{i,5}$
Fetch address	$a_{i,6}$

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

$$PE = \sum_1^n E_i + \sum_1^{n-1} O_{i,i+1} + \sum \varepsilon \tag{2}$$

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

3. PURE BASE COST AND INTER-INSTRUCTION COST MODELS – ANALYSIS OF THE RESULTS

For measuring the pure base costs, loops with instances of a reference instruction (the NOP was used) and the one test instruction were executed. The condition of the pipeline structure of the ARM7TDMI processor during the execution of such loops is shown in Table 3. The energy of the test instruction was calculated as the sum of the energy consumed in the clock cycles required for this instruction to be executed minus two times the energy budget of the NOP instruction [15]. Due to the pipeline structure, two NOP instructions are also executed in the clock cycles needed for the execution of a test instruction as it can be observed in Table 1 for *n*+1, *n*+2 and *n*+3. Complex instructions which need more cycles are modeled in the same way. They use more than three cycles (multi-cycle instructions) but only two NOP are also performed during their execution. It should be noted that, although the proposed method is applied for the ARM7TDMI, its application to processors with more pipeline stages is straightforward.

Table 3: Pipeline states during the execution of instructions for measuring base costs

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**

In the measurements for the inter-instruction costs, program loops featuring appropriate instruction pairs were formed. The condition of the pipeline states is shown in Table 4. In this case, in four clock cycles one *Instr1*, one *Instr2* and two reference instructions are executed.

Table 4: Pipeline states during the execution of instructions for measuring inter-instruction costs

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**

The complete models for the instruction-level energy consumption of the ARM7TDMI can be found in [16]. Thousand experiments corresponding to the execution of loops of instruction instances on the processor to realize the appropriate pipeline conditions were implemented. For the measurement of the instantaneous current of the processor the measuring environment proposed in [13] was employed. Pure base costs of all the instructions and for all the addressing modes are given. Some results for the pure base cost are shown in Table 5. The values of the pure base costs present most of the time a difference less than 20% in the energy of the instructions which are executed in the same number of cycles. Also, it has been shown that the existence of a condition in the instruction does not influence significantly the energy of the instruction. For the instructions which need more cycles (clocks 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.

Table 5. Base cost energy consumption (for zero operands)

Instruction	**E (nJ)**
ADD R2, R0, R1	0.910
AND R2, R0, R1	0.856
ORR R2, R0, R1	0.907
ORRS R2, R0, R1	0.967
MOV R2, R1	0.935
MOV R0, R0	0.903
ADD R2, R0, R1, ASR R3	2.137
B label	3.095
LDR R2, [R1, R3]	2.774
STR R2, [R1, R3]	1.961
MUL R2, R0, R1	2.768
MLA R2, R0, R1, R10	3.748
CMP R0, R1	0.751
SWP R2, R0, [R1]	3.917
MRS R2, CPSR	0.977
MSR CPSR_f, R2	1.143

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 of these groups. In this way we keep the size of the required model values reasonable without significant degradation of the accuracy (less than 5% in the inter-instruction cost by using only representative instructions). For example, when executing data-processing instructions, the second (flexible) operand determines which functional units shall be used for executing the instruction. Thus, we can distinguish four different categories for the data-processing instructions, which are shown in Table 6. The corresponding inter-instruction costs for the ADD instruction for different groups are given in Table 7. This concept is also applied to other instruction types, so that only a representative part of the instruction and addressing mode range has to be explicitly measured.

Table 6. Grouping of the data-processing instruction addressing modes based on architectural characteristics

Group	**2nd source operand functionality**	**Addressing modes**
1	Shift amount from a register	1, 3, 5, 7
2	Shift amount from immediate	2, 4, 6, 8, 11
3	Register operand	9
4	Immediate operand	10

Most of the values of the inter-instruction costs have negative sign as it was expected due to the followed approach [15]. The contribution of the inter-instruction costs 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. Furthermore, it has been shown that there is no symmetry in the inter-instruction cost for a pair of instructions. For example, the execution of the LDR after the ADD (in arithmetic shift right by register addressing mode) presents a cost of 0.064nJoule while the execution of ADD after LDR presents a cost of – 0.122nJoule. This seems more reasonable from the case of symmetric inter-instruction costs as Tiwari method supposes.

Table 7. Inter-instruction effect costs for representative addressing modes of the ADD instruction

Group	Group			
	1	2	3	4
1	-0.33	-0.22	-0.23	-0.16
2	-0.27	-0.18	-0.17	-0.10
3	-0.09	-0.06	-0.19	-0.20
4	-0.14	-0.05	-0.09	-0.07

To determine the accuracy of the method a number of programs with various instructions (CPI=1 or CPI>1) have been created [16]. 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 and also calculated by using the instruction-level energy models derived by the proposed method. The error was found to be up to 1.5%.

4. ANALYSIS OF THE ENERGY DEPENDENCY ON THE INSTRUCTION PARAMETERS

The dependency of the energy of the instructions on the values of the instruction parameters and the operands, called energy sensitive factors, was also studied. Energy depends on the number of 1s in the word structures of these entities. The energy-sensitive factors are the register numbers, the register values, the immediate values, the operand values, the operand addresses and the fetch addresses of the instructions. The effect of each factor was studied separately from the others since the correlation between the effect of these factors is insignificant (see Table 1).

The observed energy dependency can be approximated with sufficient accuracy by linear functions. Coefficients have been 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 accuracy.

The grouping of the instructions for the derivation of the coefficients and the corresponding measurements are presented in [17]. According to the results the linear dependency mentioned above is obvious. Some results are presented here. In Figure 1 the effect of the register number for data-processing instructions in register addressing mode is illustrated.

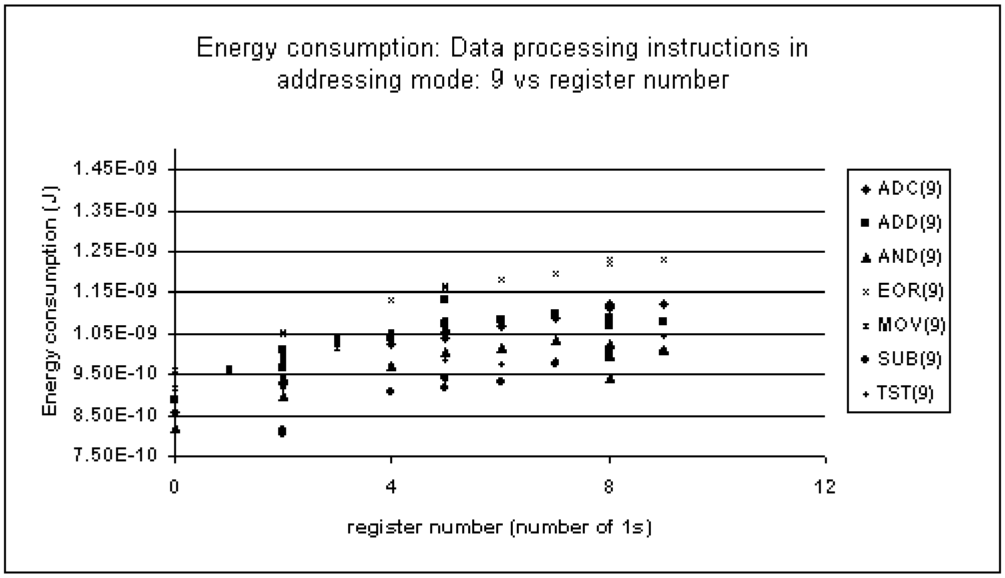

Figure 1. The effect of register number for data-processing instructions in register addressing mode

The actual physical measurements versus estimated energy values for the ADC instruction is shown in Table 8 where the achieved for the selected coefficient accuracy is also given. The error is less than 3%. Such a value of the error characterizes all the selected coefficients.

Table 8. Actual physical measurements versus estimated energy values for the ADC in register addressing mode

Num 1s	Estimated	Measured	% error	Model parameters	
0	0.874	0.855	2.20		
2	0.936	0.929	0.74		
2	0.936	0.924	1.23		
5	1.028	1.040	1.19		
6	1.059	1.067	0.77		
8	1.121	1.119	0.16	$a_{i,1}$	3.08E-02
8	1.121	1.124	0.28	b	0.874
9	1.151	1.124	2.47		
7	1.090	1.088	0.17		
5	1.028	1.054	2.46		
8	1.121	1.114	0.61		
4	0.997	1.023	2.51		

The energy consumption versus the aggregate number of 1s in the register values for the ADD instruction in the register addressing mode is given in Figure 2. The linear dependency is obvious.

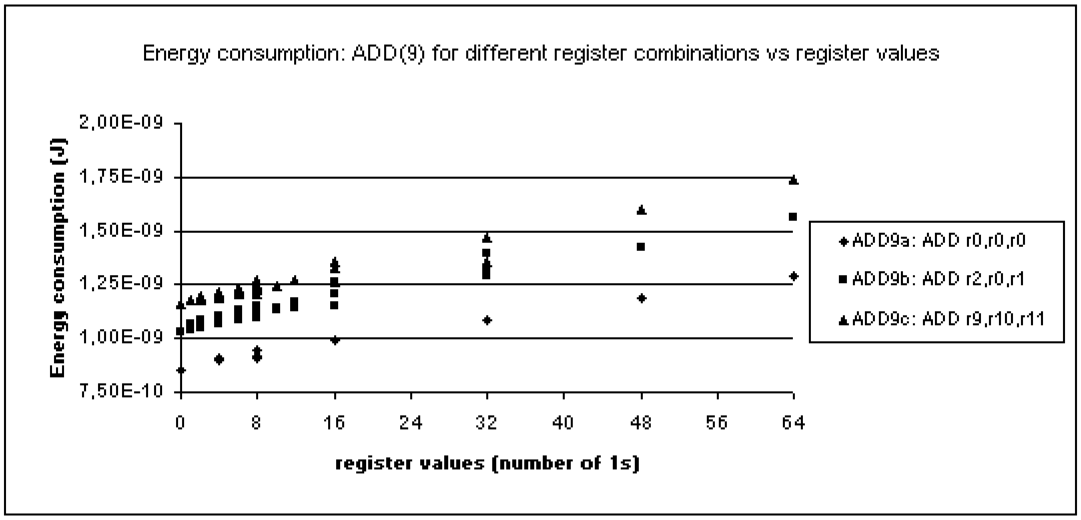

Figure 2. Energy consumption versus the aggregate number of 1s in the register values for the ADD instruction in register addressing mode

The quantities which are transferred from memory to register or inversely during the execution of load or store instructions also affect the energy of the instructions. The energy consumption versus the number of 1s in operand values for the STR instruction is shown in Figure 3.

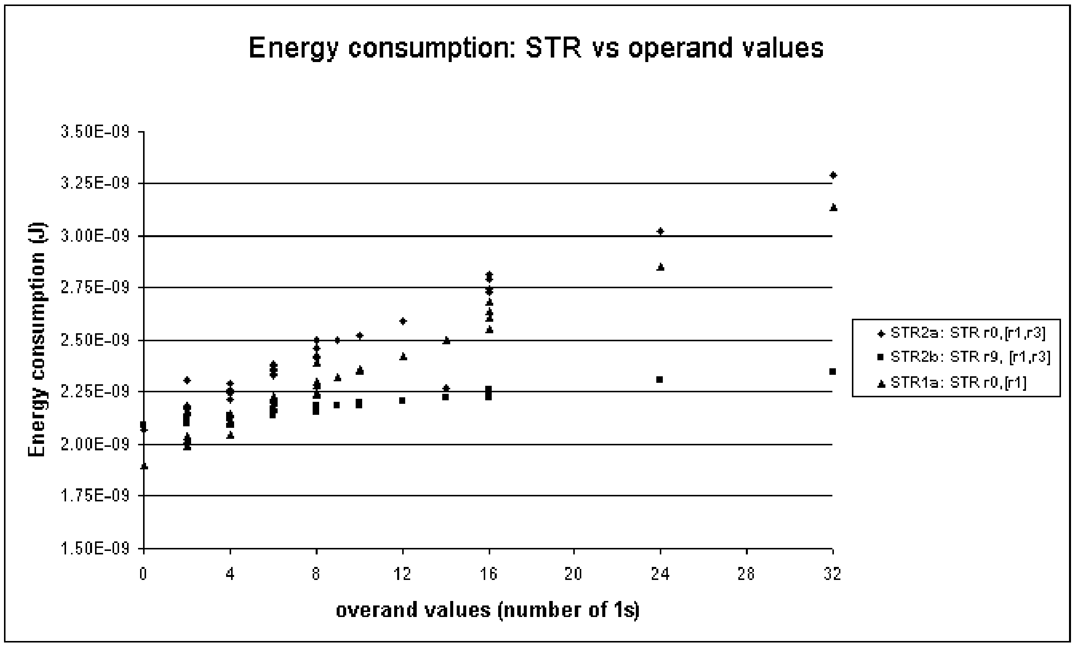

Figure 3. Energy consumption versus the number of 1s in operand values for the STR instruction

The effect of operand addresses in the energy consumption of load and store instructions versus the number of 1s is evaluated by varying the offset operand address values which corresponds to the displacement on a specified base memory address. Test measurements are given in Figure 4 and the extracted coefficients for the effect of the operand addresses are given in Table 9. The standard deviation of the coefficients from the selected ones are also given. As can be seen from the coefficient values in [17] all the sensitive-energy factors except the register values present almost equivalent effect on the energy of the instruction. The effect of the register values is one order lower except of the case of multiplications where it has similar to the others energy-sensitive factors effect.

Finally there is only a moderate dependence versus the number of 1s in the instruction fetch address and therefore the corresponding energy cost is not considered in our models.

Table 9. Coefficients ($a_{i,5}$) for the operand addresses effect on load and store instructions

Instruction type	Addressing mode group	Coefficient value	Std. Dev.
LDR	Immediate offset	4.55E-02	2.41E-03
LDR	Register offset	2.58E-02	
LDR	Scaled register offset	5.20E-03	
STR	Immediate offset	3.85E-02	8.31E-04
STR	Register offset	1.61E-02	
STR	Scaled register offset	7.27E-03	

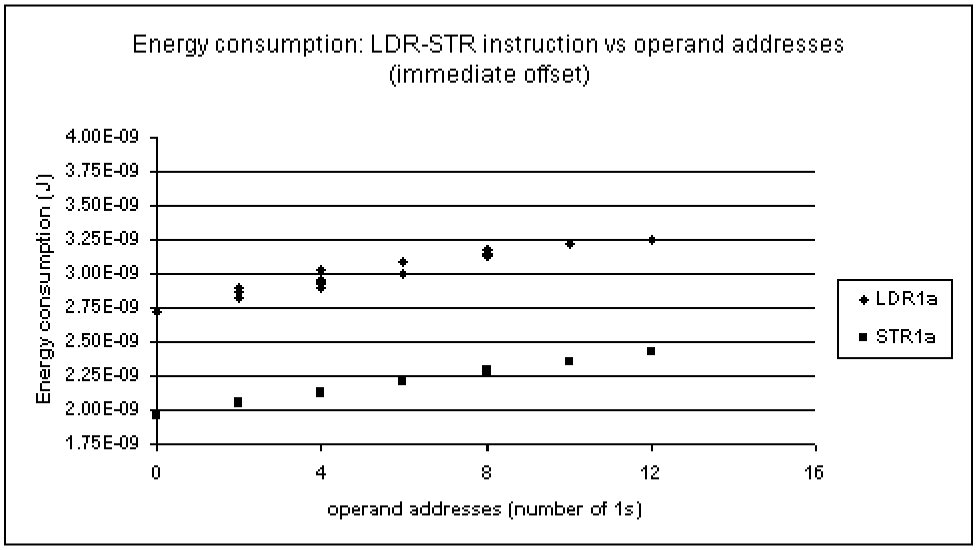

Figure 4. Energy consumption versus the number of 1s in the operand addresses for the LDR and STR instructions (immediate offset addressing mode)

5. ANALYSIS OF THE EFFECT OF PIPELINE STALLS AND FLUSHES

The ARM processor supports the RISC architectural paradigm. Moreover, the processing of the ARM7 instructions is subdivided into three pipeline stages (Instruction Fetch – Instruction Decode – Execute) to improve performance in terms of throughput.

Generally, the processing flow can be delayed due to a condition that cannot be satisfied at the time needed. When such a situation occurs, the pipeline is stalled and this irregularity results in a pipeline stage with no useful computations performed. In the case of the ARM7TDMI 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.

Small program loops to activate corresponding conditions have been implemented in order to measure the effect of these idle cycles on the energy consumption. It was found that single absolute overhead values to describe their cost in energy, are sufficient. The corresponding overhead values are shown in Table 10.

Table 10. Model parameters for the effect of pipeline stalls and flushes

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

6. MODEL ACCURACY - RESULTS

To evaluate the absolute accuracy of our modeling approach, real programs were used as benchmarks. Table 11 gives the measured and estimated energy consumption for a small program kernel. The corresponding assembly list has been extracted from a C program by utilizing the facilities of the *armcc* tool, shipped with the ARM ADS software distribution. The measured values correspond to the energy consumption during the cycles while the estimated values correspond to the estimated energy consumption of the instructions. The overall energy dissipation is calculated by summing up all the individual contributions. In Table 12 the measured and estimated energy consumption for five common software kernels are presented. According to our results the error of our approach in real life programs was found to be less than 6%.

A software framework for the estimation of the energy amount needed for the execution of a program has been developed for the ARM7TDMI processor. This software tool employs the instruction-level energy models that have been developed, enabling exploration of various alternatives of a given program, in order to optimize its energy consumption. It receives as input the trace file of executed assembly instructions and estimates the base, the inter-instruction energy cost and the effect of the instruction parameters and pipeline stalls/flushes of a program. (This tool is available at *electronics.physics.auth.gr/easy* web-site)

Table 11. Comparison of estimated and measured energy consumption of a real kernel

Instruction formation	Cycle	Measured	Ebase	Einter	Eregnum	Eregval	Eimm	Estall	Estimated
MOV a1, #5	1	0.925	0.930		0.000	0.000	0.069		0.999
SUB a1, a1, #1	2	0.943	0.800	-0.092	0.000	0.015	0.068		0.791
CMP a1, #1	3	0.811	0.830	-0.112	0.000	0.008	0.068		0.793
BGT label	4	0.768	3.100	-0.130					2.970
	5	1.022		-0.032					-0.032
	6	0.921							0.000
SUB a1, a1, #1	7	0.905	0.800		0.000	0.008	0.068		0.875
CMP a1, #1	8	0.950	0.830	-0.112	0.000	0.015	0.068		0.801
BGT label	9	0.767	3.100	-0.130					2.970
	10	1.025		-0.032					-0.032
	11	0.926							0.000
SUB a1, a1, #1	12	0.904	0.800		0.000	0.015	0.068		0.883
CMP a1, #1	13	0.944	0.830	-0.112	0.000	0.008	0.068		0.793
BGT label	14	0.768	3.100	-0.130					2.970
	15	1.023		-0.032					-0.032
	16	0.926							0.000
SUB a1, a1, #1	17	0.909	0.800		0.000	0.008	0.068		0.875
CMP a1, #1	18	0.946	0.830	-0.112	0.000	0.008	0.068		0.793
BGT label	19	0.771		-0.092				1.601	1.509
MOV a1, #0	20	1.053	0.930		0.000	0.008	0.000		0.938
	21	1.232					0.000		0.000
	22	0.938							0.000
Total		18.571							18.865
% error									-1.58

Table 12: Comparison of estimated and measured energy consumption for various real kernels

	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

Benchmark description:
1. cadd: complex addition
2. cmul: complex multiplication
3. fir: FIR filter
4. sad: sum of absolute differences (used in motion estimation algorithm of MPEG video coding)
5. ablend: alpha blending (an image compositing algorithm)

7. CONCLUSIONS

The measurements taken for the development of instruction-level energy models for the ARM7TDMI embedded processor are presented and analyzed. The instantaneous current drawn by the processor is measured and integrated in clock cycles to derive the consumed energy. Appropriate measuring environment and modeling methodology has been established for this purpose. The energy of an instruction is analyzed in three components. These components correspond to the pure base energy cost of the instruction, the inter-instruction cost and the effect of the instruction parameters. The values for these components were presented, analyzed and discussed. The proposed approach in modeling the software energy of microprocessors has been validated by the results and only up to 6% error in energy modeling was observed for real software kernels.

ACKNOWLEDGMENT

This work is performed within the project IST-2000-30093-EASY, funded by European Union's IST framework program.

REFERENCES

[1] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol. 2, No. 4, pp. 437-445, December 1994.

[2] V. Tiwari, S. Malik, A. Wolfe, and M. Tien-Chen Lee, "Instruction Level Power Analysis and Optimization of Software," Journal of VLSI Signal Processing, Vol. 13, No. 2-3, pp. 223-238, August 1996.

[3] M. Tien-Chen Lee, V. Tiwari, S. Malik, and M. Fujita, “Power Analysis and Minimization Tehniques for Embedded DSP Software,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 123-135, March 1997.

[4] V. Tiwari and T.C. Lee, “Power Analysis of a 32-bit Embedded Microcontroller,” VLSI Design Journal, Vol. 7, No. 3, 1998.

[5] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel, “An Accurate and Fine Grain Instruction-Level Energy Model supporting Software Optimizations,” in Proceedings of the International Workshop PATMOS, Yverdon-les-bains, Switzerland, September 2001.

[6] J. T. Russell, and M. F. Jacome, “Software Power Estimation and Optimization for High Performance, 32-bit Embedded Processors,” in Proceedings of the International Conference on Computer Design (ICCD ’98), Austin, TX, October 1998.

[7] N. Chang, K. Kim, and H. Gyu Lee, “Cycle-Accurate Energy Consumption Measurement and Analysis: Case Study of ARM7TDMI,” IEEE Transactions on VLSI Systems, Vol. 10, No. 2, pp. 146-154, April 2002.

[8] H. Mehta, R.M. Owens, and M. J. Irwin, “Instruction level power profiling,” In Proceedings of the Inter. Conference on Accoustics, Speech and Signal Processing, 1996.

[9] R.Y.Chen, M.J. Irwin, and R.S. Bajwa, “ An architectural level power estimator,” in proceedings of the Power-Driven Microarchitecture Workshop, 1998.

[10] W. Ye, N. Vijaykrishnam, M. Kandemir, and M. J. Irwin, “ The design and use of SimplePower: A cycle-accurate energy estimation tool”, in proceedings of the Annual ACM IEEE Design Automation Conference, pp. 340-345, June 2000.

[11] T. Stouraitis, SOFLOPO, Low Power Development for Embedded Applications, Esprit project, Deliverable 2.2: Physical measurements, University of Patras, December 1998.

[12] G. Sinevriotis, T. Stouraitis, “Power Analysis of the ARM7 Embedded Microprocessor”, in Proc. of *International Workshop on Power and Timing Modeling, Optimization and Simulation* (PATMOS’99), pp. 261-270, Kos, Greece, Oct. 1999.

[13] 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, Vol 52, No 4, pp. 1206-1212, Aug. 2003.

[14] 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, Sept. 2002

[15] S. Nikolaidis, N. Kavvadias, T. Laopoulos, L. Bisdounis and S. Blionas, "Instruction Level Energy Modeling for Pipelined Processors," Int. Workshop on Power and Timing Modeling, Optimization and Simulation, Turin, Italy, Sept. 2003

[16] S. Nikolaidis, N. Kavvadias, and P. Neofotistos, "Instruction level power measurements and analysis", IST-2000-30093/EASY Project, Deliverable 15, Sept 2002.http://electronics.physics.auth.gr/easy

[17] S. Nikolaidis, N. Kavvadias, and P. Neofotistos, "Instruction level power models for embedded processors", IST-2000-30093/EASY Project, Deliverable 21, Dec 2002. http:// electronics.physics.auth.gr/easy