Reprint

SOFLOPO: Towards Systematic Software Exploitation for Low-Power Designs

G. Sinevriotis, A. Leventis, D. Anastasiadou, C. Stavroulopoulos, T.Papadopoulos, T. Antonakopoulos and T. Stouraitis

International Symposium on Low Power Electronics and Design- ISLPED'00

PALACE RAPALLO/PORTOFINO COAST, ITALY, JULY 26-27, 2000

SOFLOPO: Towards Systematic Software Exploitation for Low-Power Designs

Giannis Sinevriotis[1], Apostolos Leventis[2], Despina Anastasiadou[3], Chris Stavroulopoulos[4], Thanasis Papadopoulos[5], Theodore Antonakopoulos[6], Thanos Stouraitis[7]

[1]Electrical Engineering and Computer Technology Dept,
University of Patras, Rio 26500, Greece
synevrio@ee.upatras.gr
[2,3,4,5,6]Laboratory of Electromagnetics
Electrical Engineering and Computer Technology Dept,
University of Patras, Rio 26500, Greece
{leventis,desan,cstavr,thanasis,antonako}@loe.ee.upatras.gr
[7] Electrical Engineering and Computer Technology Dept,
University of Patras, Rio 26500, Greece
thanos@ee.upatras.gr

Abstract. The SOFLOPO project has dealt with the software-related energy-optimization for embedded applications. Based upon the fact that software has a conceivable impact upon the total energy dissipation, we tried to minimize this impact, using assembly-level code transformations. The instruction costs were obtained through the tried-and-tested methodology of physical measurements. Two target processors have been selected: the ARM7TDMI RISC processor and the Motorola DSP56156 DSP processor. The obtained results were verified upon a commercial implementation of the IEEE 802.11 wireless multimedia protocol.

1 Introduction

An ever-increasing number of digital systems in electronics industry are implemented as embedded applications. Embedded applications are characterized by the splitting of their functionality between application-specific circuits and application-specific software running on a dedicated processor. A big portion of embedded applications is implemented as parts of mobile, battery-operated systems; therefore they are power-critical. Design for low-power has thus become increasingly important, driving the need for optimizing every part of such systems.

The software can have a big impact on the power dissipation. This is indicated by the fact that the battery life of mobile systems, such as portable personal computers, can vary depending on the software they run. Traditionally, costly simulation of detailed processor models is needed in order to evaluate the power dissipation of a processor as it runs some specific software. In most cases, such models are not available to third parties.

Another option is to evaluate the power dissipation through physical means [4]. The idea is to measure the current that is drawn by a processor as it executes some software; upon these measurements *instruction costs* that relate energy dissipation and software execution are established. The current drawn by the microprocessor as it executes a specific instruction is a measure of the energy cost of the instruction and thus of the total energy required to execute a program. The average power P consumed by a processor while running a certain program is given by: $P = I \times V$, where I is the average current and V the supply voltage. Therefore, the energy E *co*nsumed by a program is given by: $E = I \times T$, where T is the execution time of the program. The energy cost of an individual instruction becomes $E = I \times V \times N \times t$, where N the required number of clock cycles and t the clock period. Since V, N, t constant, the consumed energy the consumed energy is a function of the drawn current.

2 Instruction Level Power Models

It has been shown that the total energy cost of a program cannot be calculated by the summation of the energy costs of the individual instructions [4-7]. In real programs, running on real processors, there are other effects that have an impact on the total energy cost, such as the effect of circuit state and pipeline stalls. These effects have also to be taken into account in order to establish accurate instruction-level power models. The components of these power models are:

- *Basic Costs*. These are the costs that are associated with the basic processing required to execute the instruction. To determine these costs, infinite loops of a given instruction are formed and fed to the processor. The construction of these loops should guarantee that no undesired effects such as pipeline stalls take place.
- *Inter-Instruction Effects*. The switching activity in a circuit, and therefore the power consumption it implies, results from the change in two consecutive sets of inputs. The change in circuit state is bigger between different instructions. For sequential circuits, the effect of circuit state can expand to many instructions; it has been shown however that is suffices to examine pair of instructions [4],[7]. To measure the impact of the change in circuit state the one instruction loops used for the measurement of basic costs are replaced by sequences of alternating instructions. The effect of circuit state is determined as the current value measured minus the average base cost of the two instructions.
- *Other effect*s. These cover all the other processor-specific effects that may affect the energy dissipation. These include the cost of cache misses (when external memories are taken into account) and pipeline stalls. Special instruction sequences that isolate the impact of such effects have to be written and executed.

Taking these costs into account, the overall energy cost E_p of a program P is given by Equation (1).

$$E_p = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) + \sum_k E_k \quad \textbf{(1)}$$

Where B_i the base cost of the instruction i, N_i the number of the occurrences of the instruction i, $O_{i,j}$ the circuit state overhead from the instruction pair i, j and E_k the energy costs of other effects. The aforementioned formula provides quite accurate results in practice [4],[7].

3 Processor Overview

3.1 The ARM7 Processor

The ARM 7TDMI processor core is widely used in embedded applications. Its key strengths are the small die size combined with market-leading MIPS/mW performance. Also, its open architecture makes it the obvious choice for the embedded system designer. A full description of the ARM7 family can be found in various sources [2].

The instruction set can be divided into three broad classes of instructions that can be subdivided into subclasses:

- Branch instructions.
- Data-processing instructions.
    - Data-processing instructions proper.
    - Multiply instructions.
    - Status register transfer instructions.
- Load and store instructions.
    - Load and store single register value instructions.
    - Load and store multiple register value instructions.
    - Swap a register value with the value of a memory location instruction.

In addition, every instruction supports multiple addressing modes. Up to 21 distinct addressing modes may be supported for a single instruction.

3.2 The DSP Processor

On a single semiconductor chip, the DSP56156 comprises a very efficient 16-bit digital signal processing core, program and data memories, a number of peripherals, and system support circuitry. This combination of features makes the DSP56156 a cost-effective, high-performance solution for many DSP applications, especially speech coding, digital communications, and cellular applications. The central processing unit of the DSP56156 is the DSP56100 core processor. Like all DSP56100- based DSPs, the DSP56156 consists of three execution units operating in parallel, allowing up to six operations to be performed during each instruction cycle.

The instruction set is divided into the following groups:

- Arithmetic
- Logical
- Bit Field Manipulation

- Loop
- Move
- Program Control

4 The Experimental Method

In order to obtain the required instruction costs, a complete laboratory environment was created. The boards were designed for current measurements and thus the power supply connection to the CPU core was isolated from the rest of the system. The measuring environment also includes the required power supplies as well as a high-accuracy ammeter. The whole environment has been integrated into a host PC. It is depicted in Figure 1, for the case of the ARM7 processor. A similar setup was used to evaluate the instruction costs of the DSP processor.

For the case of the DSP processor, a complete board was constructed, so as to be able to isolate the 56100 processor core from its peripherals. The board is shown in Fig 2.

We adopt the following methodology in order to gain useful current measurements. The sequence of tasks is as follows:

- Formulating a sequence of instructions
- Assembling the sequence and downloading it to the target processor
- Running the sequence and taking current measurements
- Transmitting the current measurements to a computer and processing the results in a worksheet

Special care has to be taken in the formation of the instruction sequences so that unwanted side effects, such as inter-instruction dependence that causes pipeline stalls and thus alter the obtained results, are avoided. Also, the registers should be initialized at the beginning of the sequence, because, as it will be shown later, the value of the instruction operands may have a significant impact upon the power dissipation.

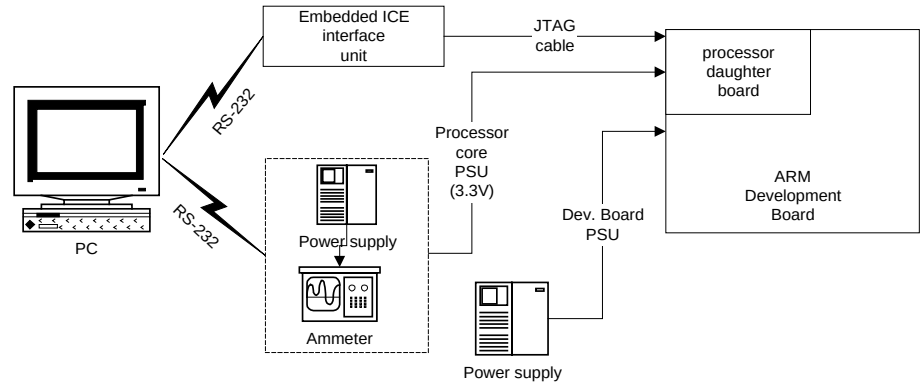

Fig. 1. The ARM7 laboratory setup

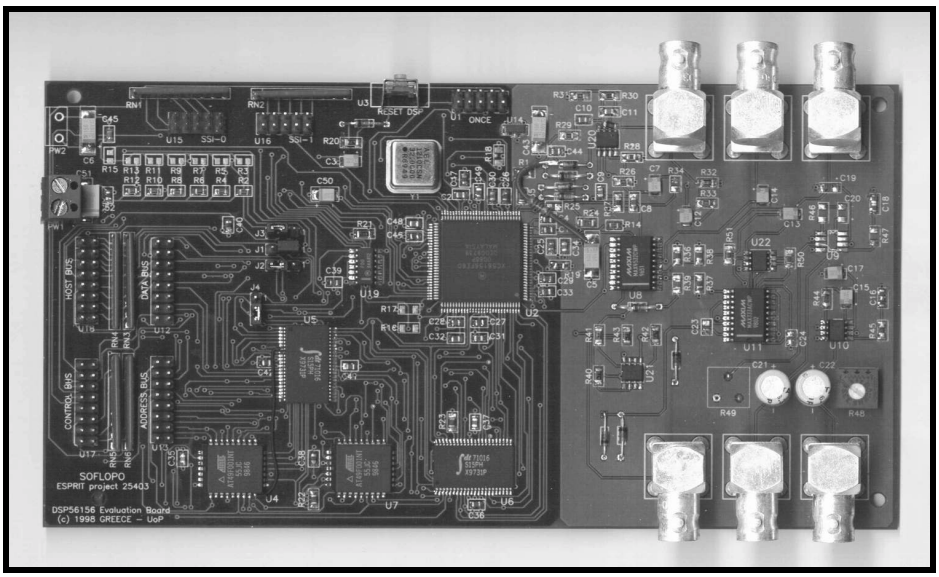

Fig. 2. The specially designed board for the current measurements on the Motorola DSP56100 processor core

4.1 The ARM7 Measurements

4.1.1 Basic Costs

A series of measurements was made to determine the basic costs of the ARM instructions. ARM instructions can be categorized with respect to the addressing modes they support. A selection of the basic costs of data processing instructions for all possible addressing modes is shown in Table 1 (All values are in mA). The instructions are of the form *INSTR r0,r1,[r2],[#r3]* , where "[]" denotes an optional operator. It should be noted that the register file was found to be a symmetrical one; the choice of the specific registers does not affect the power dissipation. Also, all register values where set to 0, in the loop header. In all, about 500 basic instruction costs were measured.

It can be noted that there exist sets of instructions that exhibit quite similar power dissipation characteristics for all the addressing modes. Also, it can be easily noticed that the addressing modes that include shifting of the result by the contents of the register r3, are more energy consuming than the rest addressing modes. Moreover, these addressing modes take two cycles to execute, instead of one.

Similar analysis has taken place for the rest of the instructions. Some of the basic costs measured are shown in Table 2.

The data values in the registers affect power dissipation. To demonstrate this, a series of measurements were made in which varying data values where set to the registers. The results are shown in Table 3. It can be shown that the power dissipation increases with the number of 1's in the operands.

Addressing Mode	Instruction						
	ADC	AND	BIC	EOR	MOV	SUB	TST
ASR BY REGISTER	10.68	10.77	10.02	10.39	11.20	9.52	10.73
ASR IMMEDIATE	7.17	7.05	5.71	8.07	8.00	5.74	6.95
ROR REGISTER	10.69	10.77	10.02	10.39	11.21	9.52	10.73
ROR IMMEDIATE	7.24	7.12	5.77	8.14	8.06	5.82	7.02
LSL BY REGISTER	10.81	10.79	10.04	10.42	11.23	9.55	10.76
LSL IMMEDIATE	7.15	7.03	5.67	8.04	7.95	5.72	6.92
LSR BY REGISTER	10.68	10.76	10.01	10.39	11.20	9.52	10.73
LSR IMMEDIATE	7.20	7.08	5.73	8.10	8.01	5.78	6.98
REGISTER	7.00	6.88	5.52	7.89	7.80	5.57	6.76
IMMEDIATE	7.20	7.01	5.86	7.94	8.01	7.00	6.89
RPX	7.06	6.95	5.60	7.93	7.95	5.71	6.83

Table 1. Examples of basic costs of data processing instructions

Instruction	Addressing Mode	Clock Cycles	Current (mA)
MUL	-	2-5	13.75
MLA	-	3-6	13.44
LDR	IMMEDIATE	3	10.76
	OFFSET.REGISTER	3	10.95
	REGISTER OFFSET, SCALED ASR	3	10.92
	PRE-INDEXED OFFSET, REGISTER	3	9.79
STR	IMMEDIATE	2	8.55
	OFFSET.REGISTER	2	8.19
	REGISTER OFFSET, SCALED ASR	2	8.57
	PRE-INDEXED OFFSET, REGISTER	2	6.72

Table 2. Examples of basic costs of miscellaneous ARM7 instructions

Operand 1	Operand 2	Number of 1s	Current (mA)
0	1	1	7.68
0	7	3	7.83
0	63	6	8.03
0	255	8	8.17
0	511	9	8.25
0	65535	16	8.63
0	16777715	24	9.13
255	16777715	32	9.53
65535	16777715	48	9.82

Table 3. Basic costs for the ADD instruction with varying operand values.

4.1.2 Inter-instruction effects

A complete examination of all the inter-instruction effects would require writing the appropriate instruction sequences for all possible pairs of instructions. Since about 500 single instructions were measured, this would require about $500^2 = 250000$ instruction sequences. To cope with the complexity of the problem, the instructions were grouped in sets according to the processor resources they utilize. A bottom-up approach is taken; firstly possible subgroups are formed according to common addressing modes and close similarity of he measured basic costs. The inter-instruction-effects are measured for the instructions belonging in the same group; then pairs of instructions belonging to different groups are considered. It was found that the inter-instruction effects when different instructions from same groups are used are quite close. This can be justified by the fact that instructions within the same

group utilize similar processor resources; therefore switching between them represents quite similar circuit switching activity. This approach has been validated in practice using random tests.

Some indicative measurements of the inter-instruction effects are shown in the Table 4.

Instruction 1	Instruction 2	Measured Current (mA)	Overhead (mA)
ADD ASR by Reg	ADD Immediate	8.61	0.88
EOR ASR by Imm	EOR LSR by Imm	10.45	1.05
ADD Register	EOR Register	8.19	0.39
ADD ASR by Reg	EOR ROR by Imm	10.64	1.19
MUL	ADD Immediate	12.80	1.13
LDR Offset Imm.	ADD Immediate	10.33	1.09
STR Offset Imm.	SUB Immediate	8.85	1.24

Table 4. Examples of inter-instruction effects

4.2 The DSP Measurements

In a similar fashion to the ARM instructions, the instruction costs of the Motorola DSP processor were also evaluated. The formulation of the instruction sequences takes into account the architectural features of the DSP such as the parallel processing capabilities. Initially we will compute the basic cost of each instruction by formulating instruction sequences for each DSP instruction.

The DSP architecture can be viewed as three functional units operating in parallel (Data Arithmetic Logic Unit, Address Generation Unit and Program Control Unit). The goal of the instruction set is to keep each of these units busy each instruction cycle. This achieves maximum speed and minimum use of program memory.

Most members of the Arithmetic and Logical groups of instructions allow parallel data transfers. Thus, different sequences are formulated, which also include these transfers. In this case, for the instructions which allow these transfers we formulate sequences two different types of sequences for these instructions- one type without the parallel transfers and one with these transfers. The bit field manipulation does not allow parallel transfers so sequences that measure their current consumption are formulated. The size of the loops depends on the available SRAM on the development board. The initial size has been set at 10 K program words but the number of instructions in each loop depends on the number of operands of each instruction. We have the option of adding further memory to the development board if this is required.

4.3.1 Basic Costs

The Motorola DSP 56156 consists of 53 instructions. Of these, 35 allow simultaneous parallel moves. Indicative values for the basic costs of these instructions are shown in Table 5. As in the case of the ARM processor, the register file was found to be a symmetrical one, i.e. the choice of registers has no impact upon the power dissipation. Moreover, the effect of the register operands was found to be insignificant compared to the basic costs. This is due to the fact that the DSP architecture is significantly bigger than this of the RISC processor.

Type of Instructions	Instruction	Measured Current (mA)
Instr D	abs a	70.16
	asl a	70.21
	Lsl a	70.07
	zero a	69.02
Instr S,D	add a,b	76.85
	and x0,a	70.16
	sbc x,a	74.67
	div x0,a	75.04
Bitfield Instructions	bfchg #$0310,x0	81.50
	bfclr #$0310,x:$10	80.69
	bftsth #$0310,x:$10	79.82
	bftsth #$0310,x0	72.77
Instr S1,S2,D	mpy +x0,x0,a	70.14
	mpy –y0,a1,b	69.89
	mac +x0,x0,b	77.03
	imac x1,x0,a	71.74
	macuu y0,x0,a	73.91
Move	move b,a	70.65
	move x:(r2)+,a	78.80
Instr D with parallel moves	asl a (r1)+n1	69.43
	rol a a,b	74.29
	asl a x:(r2+$12),b	84.18
Instr S, D with parallel moves	add x0,a a,b	73.46
	or x0,a a,b	69.73
	cmp x0,a a,b	69.78
Special Instructions	bra	73.55
	nop	70.14

Table 5. Examples of DSP base instruction costs

4.3.2 Inter-Instruction Effects

Contrary to the ARM processor, the effect of the circuit state in power dissipation is rather limited, as a result of the bigger circuit area of the DSP chips. Nonetheless, the

inter-instruction effects for the DSP processor were also measured, albeit in a more limited scale. In a series of measurements, the inter-instruction effect overhead is fairly constant; thus this case can be treated as CISC processors described in [4].

5 Software Power Modeling

In order to facilitate the utilization of the instruction measurements by the software optimizing tools, power dissipation models were developed. These include:

5.1 Grouping of Instructions

This essentially consists of the grouping of instructions that exhibit common power characteristics so as to simplify the optimization algorithms. As can be seen from the Table 1 the ADC, AND, TST instructions have essentially the same power dissipation characteristics. Moreover, this is also true for the inter-instruction effects of these instructions. Groups of instructions that exhibit such similarities are shown in Table 6.[1]

Grouping	Instructions
1	ADC, TST, AND, CMN, EOR, TQT, ADD, TEQ
2	MOV
3	BIC, CMP, SBC, SUB
4	RSB, RSC
5	STR, STRB
6	LDR, LDRB
7	LDRSB, LDRH, LDRSH

Table 6. Examples of instruction groupings

5.2 Operand Value Overhead

There is a strong relationship between the number of 1's in the instruction operands and the power dissipation. This is captured in Table 7 for the ALU instructions. Similar tables have been created for other types of instructions.

Number of 1's	3	6	8	16	24	32	48
Percent Overhead	1.2	4.6	6.4	12.4	18.9	24.1	27.8

Table 7. Percent power dissipation overhead according the number of 1's (indicative values)

[1] The complete set of the instruction grouping can be found in the project deliverables

5.3 Grouping of Inter-Instruction Effects

The percentage of the energy dissipation overhead due to the inter-instruction effects shows small variations between certain groups of instructions. This happens due to the fact that certain instructions utilize the same resources; thus the circuit switching activity between these instructions has very similar value. For the purpose of the analysis, it suffices to assume some constant overhead for these cases. Examples of such groupings of inter-instruction effects are shown in Tables 8,9.[2]

Instruction Grouping	1	2	3	4	5
1	0.8-2.5	-	-	-	-
2	9.5-11.5	6.5-7.5	-	-	-
3	9.5-11.5	7.5-9.5	-	4.5	5
4	9.5-11.5	10.5-15	-	-	-
5	8-10	6.5-8.5	-	-	-

Table 8. Examples of groupings of the inter-instruction effects for different addressing modes of the same instruction. The table refers to all data-processing instructions. Groupings represent different addressing modes. Groups 1,2 consists of 4 addresing modes (entries 1,3,5,7 and 2,4,6,8 in Table 1respectively) while 3,4,5 of a single addresing mode (entries 9,10,11 in Table 1). The entries consist of the variation of the percentage overhead current.

Instruction Grouping	Same Group	Different Group
ASR by Register	0.25	6.00
ASR by Immediate	1.70	15.00
ROR by Register	0.50	7.00
ROR by Immediate	1.55	15.00
LSL by Register	0.20	7.00
LSL by Immediate	1.50	17.13
LSR by Register	0.25	7.00
LSR by Immediate	1.75	15.00
Register	2.00	12.00
Immediate	2.00	12.00
RPX	2.00	10.00

Table 9. Examples of grouping of inter-instruction effects for the same different data proccesing instructions with common addressing modes. Table entries consist of the percentage overhead used in the optimization process

[2] The complete set tables for the grouping of the inter-instruction effects can be found in the project deliverables.

6 Software Energy Optimization

The results of the above were included in a software tool for the automatic energy optimization of ARM7 code. The input of the tool is the non-optimized assembly code. It works in a block-by-block basis in a multi-pass way. The steps taken are:

1. The code is split into basic blocks, i.e. continuous code fragments without branches. This ensures the required locality for the code transformations (See Figure 3).
2. The energy cost of the non-optimized code is evaluated by utilizing the power models referred in Section 5.
3. The code is optimized in a block-by-block fashion using code transformations (see Section 6.1). The energy cost of the optimized code is evaluated using the same principles as in step 2. This is repeated until no more improvements are possible.

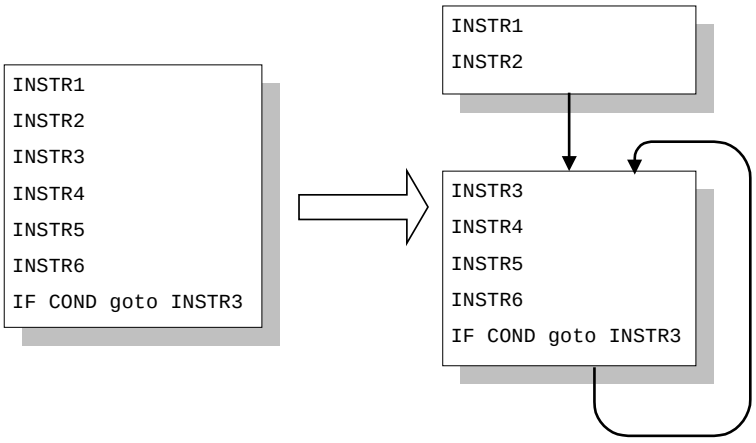

Fig. 3. Code partitioning into basic blocks

6.1 Low-Power Code Transformations

A number of transformations were employed to decrease the power dissipation of the code. The main aim of analyzing the power characteristics of the ARM7 embedded processor is to gain information in order to drive the software optimization process. Unfortunately, most of the methods proposed in literature [8],[9], are basically run-time optimization techniques, that consequently have an advantageous impact on energy optimization. Thus the number of the applicable code transformations is limited. These include:

6.1.1 Instruction Strength Reduction

It is possible to replace an expensive operator with a functionally equivalent cheaper one, in terms of energy dissipation [1]. Such an example is the multiplication *i := i*2* that can be replaced by the left shifting *i := i<<1*. The possibility of such

transformations depends on the instruction set of the target machine: a rich instruction set may provide more alternatives for strength reduction transformations.

For the ARM instruction set, such possible transformations are shown in Table 10[3].

Initial Instruction	Alternative Instruction
MUL Rx, Rm, Rn (Rn>Rm)	MUL Rx, Rn, Rm
ADD Rx, Rm, *#VAL*	SUB Rx, Rm, *-(#VAL)*
CMP Rm, *#VAL*	CMN Rm, *-(#VAL)*

Table 10. Examples of code transformations utilized in the SOFLOPO project. *#VAL* denotes an immediate value

6.1.2 Software Scheduling for Low-Power

The circuit state overhead cost can vary significantly across different instruction pairs. It is possible to reduce the power cost of a program by an appropriate ordering of instructions. This can be implemented in a systematic way, by the modification of the existing software scheduling algorithms [1],[3]. This looks up the overhead cost tables and chooses a good schedule that does not violate data dependencies. It can be implemented with any scheduling algorithm; the list-scheduling algorithm is an obvious choice. Instead of selecting the next instruction from the ready set by the assigned priorities, the next instruction is selected upon the minimization of the overhead cost.

An example of the results of a software-scheduling algorithm is shown in Table 11.

Non-Optimized Code		Optimized Code	
`CMP Ra,Rb`	**6.60**	`CMP Ra,Rb`	**6.60**
	1.32		1.32
`SUBCS Ra,Ra,Rb`	**5.63**	`SUBCS Ra,Ra,Rb`	**5.63**
	2.74		1.25
`ADDC Rc,Rd,Re,LSR #1`	**8.57**	`ADD Re,Ra,#val`	**8.55**
	2.65		2.65
`ADD Re,Ra,#val`	**8.55**	`ADDC Rc,Rd,Re,LSR #1`	**8.57**
	2.54		1.24
`SUBC Rc,#val,LSR #3`	**7.72**	`SUBC Rc,#val,LSR#3`	**7.72**
	1.80		1.80
`BNE Lbl`	**7.50**	`BNE Lbl`	**7.50**
Average Instruction Cost	9.27	Average Instruction Cost	8.80

Table 11. Example of software scheduling for low-power. Values in bold typeface denote basic costs; values in plain format denote inter-instruction effects. All values in mA

[3] The full range of the employed transformations can be found in the project deliverables

7. Obtained Results

The produced software was tested upon the implementation of the IEEE 802.11 wireless multimedia protocol. The energy dissipation of the unoptimized code was measured; this was compared against the energy dissipation of the optimized code produced with the application of the developed software. The results have shown energy savings in the region of 10%.

The fact that the energy dissipation of *only* the processor core was measured has its limitations on the achieved results. If one considers complete embedded systems the savings are expected to be significantly larger, as a result of the big impact that the memory accesses have on the system's energy dissipation. Having said that, measuring the energy dissipation of only the processor core increases the reusability of our results.

References

1. Alfred V.Aho, Ravi Sethi, Jeffrey D. Ullman, "*Compilers: Principles, Techniques and Tools*", Addison-Wesley, 1986.
2. Dave Jaggar, "Advanced RISC Machines Architectural Reference Manual", Prentice-Hall, 1996.
3. John L. Hennesy, David A. Patterson, "*Computer Architecture: A Quantitative Approach*" , Morgan-Kauffman Publishers, 1996.
4. Vivek Tiwari, Sharad Malik, Andrew Wolfe, "*Power Analysis of Embedded Software: A First Step Towards Software Power Minimization*", IEEE Transanctions on VLSI Systems, Vol. 2, No. 4, December 1994 pp. 437-445.
5. Mike Tien-Chien Lee, Vivek Tiwari, Sharad Malik, Masahiro Fujita, "*Power Analysis and Minimization Techniques for Embedded DSP Software*", IEEE Transanctions on VLSI Systems, Vol. 5, No. 5, March 1997 pp. 123-135.
6. Vivek Tiwari, Sharad Malik, Andrew Wolfe, "*Instruction Level Power Analysis and Optimization of Software*", Journal of VLSI Signal Processing, 1-18(1996), Kluwer Academic Press.
7. Vivek Tiwari, Mike Tien-Chien Lee, "*Power Analysis of a 32-bit Embedded Microcontroller*", VLSI Design Journal, Vol. 7, No. 3, 1998
8. Vivek Tiwari, Sharad Malik, Andrew Wolfe, "*Compilation Techniques for Low Energy: An Oveview*", Procceding of the 1994 IEEE Symposium on Low Power Electronics, pp. 38-39.
9. Ping-Wen Ong, Ran-Hong Yan, "*Power-Conscious Software Design – a Framework for Modelling Software on Hardware"* Procceding of the 1994 IEEE Symposium on Low Power Electronics, pp. 36-37.
10. Giannis Sinevriotis and Thanos Stouraitis, "*Power Analysis of the ARM7 Embedded Microprocessor"*, Procceding of the PATMOS'99 Conference, pp. 261-270