ENERGY COMPLEXITY OF SOFTWARE IN EMBEDDED SYSTEMS

Kostas Zotos, Andreas Litke, Alexander Chatzigeorgiou, Spyros Nikolaidis[1], George Stephanides
Dept. of Applied Informatics, University of Macedonia, 54006 Thessaloniki, Greece
[1]Dept. of Physics, Aristotle University of Thessaloniki, Thessaloniki, Greece
(zotos@uom.gr, litke@uom.gr, achat@uom.gr, snikolaid@physics.auth.gr, steph@uom.gr)

ABSTRACT
The importance of low power consumption is widely acknowledged due to the increasing use of portable devices, which require minimizing the consumption of energy. The energy in a computational system depends heavily on the software being executed, since it determines the activity in the underlying circuitry. In this paper we introduce the notion of *energy complexity* of an algorithm for estimating the required energy consumption. As test vehicle we employ matrix multiplication algorithms and from the results it can be observed that energy complexity in combination with computational complexity, provides an accurate estimation for the energy consumed in the system.

KEY WORDS
software design and implementation, computational complexity, low-power design, energy estimation

1. Introduction

The vast majority of microprocessors being produced today are incorporated in embedded systems, which are mainly included in portable devices. The later ones require the lowest power operation achievable, since they rely on batteries for power supply. Furthermore, high power consumption raises other important issues, such as the cost associated with cooling the system, due to the generated heat, as well as reliability concerns. A lot of optimization efforts have been devoted to restructuring the hardware used, to decrease power consumption. However, recent research has proved that software is the dominant factor in the power consumption of a computing system [1], since it determines the number of energy consuming "0" to "1" transitions in the underlying digital circuitry. Here, we attempt, in analogy to the well established computational complexity (time complexity), to estimate the ***energy complexity*** for a given algorithm, as a means to characterize the expected energy consumption.

2. Energy Consumption

To clarify the reasons why energy consumption of a program varies, it is necessary to name the main sources of power consumption in a simplified model of an embedded system. System power falls into mainly two categories, each of which is described next.

2.1 Processor Power

When instructions are fetched, decoded or executed in the processor, the nodes in the underlying CMOS digital circuits switch states. Without getting into details, the dynamic power dissipation for a digital system (the power consumed during the switching of gates) is given by [2]:

$$P_{avg} = a \cdot C_L \cdot V_{DD}^2 \cdot f_{clk}$$

where C_L is the total capacitance being switched, V_{DD} is the supply voltage, f_{clk} is the clock frequency, and α is the node transition activity factor. This factor essentially represents the average number of times a given node in the circuit makes a power consuming transition (from logic "low" to logic "high"). For a computational system, this factor is primarily determined by the executed software, which dictates the operation (signal transitions) in the circuit.
For any computing system, the switching activity associated with the execution of instructions in the processing unit, constitutes the so-called base energy cost [1]. The change in circuit state between consecutive instructions is captured by the overhead or inter-instruction cost. To calculate the total energy, which is dissipated, all that is needed is to sum up all base and overhead costs for a given program.

2.2 Memory power

We assume that the system architecture consists of two memories, namely the instruction memory and data memory (Harvard architecture) [3] and that there is no cache memory. The energy consumption of the instruction memory depends on the code size and on the number of executed instructions that correspond to instruction fetches, whereas that of the data memory depends on the volume of data being processed by the application and on how often the later accesses data.

3. Energy Complexity

Any computational complexity measure is related to the running time [4]. The running time of an algorithm on a particular input is the number of primitive operations or "steps" executed. We shall assume a generic one-processor, random-access machine (RAM) model of computation as our implementation technology and understand that our algorithms will be implemented as computer programs. In the RAM model, instructions are executed one after another, with no concurrent operations [4]. It is usually assumed that a constant amount of time is required to execute each line code. This viewpoint is keeping with the RAM model, and also reflects how the pseudocode would be implemented on most actual computers. The RAM model contains instructions commonly found in real computers: arithmetic (add, subtract, multiply, remainder, floor, ceiling), data movement (load, store, copy), and control (conditional and unconditional branch, subroutine call and return). Each such instruction takes a constant amount of time.

In analogy to the computational complexity as described in [4] the energy complexity of an algorithm could be used to characterize the energy dissipation. The aim here is to extract a polynomial expression of the number of data memory accesses in addition to the expressions of the number of executed primitive operations or instructions. As a result, the resources of interest are not time or space but rather energy dissipation. In the following examples we do not employ the Big O notation so as to reveal even small differences between the algorithms under study.

In this way it is possible to come up with a polynomial that represents the number of accesses to the data memory. Since each access to a memory has a known (measurable) energy cost [5] the data memory energy consumption can be easily calculated.

On the other hand, polynomial expressions of the number of primitive operations are suitable for extracting the energy consumed within the processor and in the instruction memory. Since each primitive operation maps approximately to an independent assembly instruction whose energy consumption can be measured [6] and since the number of accesses to the instruction memory are equal to the executed assembly instructions, these two energy components can be easily obtained.

Thus, the total energy complexity would be calculated as:

$$E_{total} = c_1 \cdot E_{proc} + c_2 \cdot E_{instr\_mem} + c_3 \cdot E_{data\_mem}$$

where:

$c_1 \cdot E_{proc}$ corresponds to the processor energy and is a polynomial expression of the number of primitive operations times a coefficient c_1. Coefficient c_1 corresponds to the average energy consumed during the execution of an assembly instruction and can be accurately estimated from physical power measurements [6] and profiling of representative applications.

$c_2 \cdot E_{instr\_mem}$ corresponds to the instruction memory energy and is a polynomial expression of the number of primitive operations times a coefficient c_2. Coefficient c_2 corresponds to the energy cost of an access to the instruction memory.

$c_3 \cdot E_{data\_mem}$ corresponds to the data memory energy and is a polynomial expression of the number of memory accesses times a coefficient c_3. Coefficient c_3 corresponds to the energy cost of an access to the data memory.

The coefficients c_1, c_2 and c_3 are dependent on the computer system used and E_{total} is the total calculated energy complexity of the algorithm under study. However, a calibration for the corresponding coefficient values should be performed for each target platform.

4. Framework Setup and Results

To evaluate the proposed energy complexity, a generalized target architecture was considered (Fig.1). It is based on the ARM7 integer processor core [7], which is widely used in embedded applications due to its promising MIPS/mW performance [8]. The process that has been followed during the conduction of the aforementioned experiments (Fig. 2) begins with the compilation of each C++ code with the use of the compiler of the ARM Developer Suite [9]. At this stage, we were able to obtain the code size. Next and after the debugging, a trace file was produced which logged instructions and memory accesses. The debugger provided the total number of cycles. A profiler was specially developed for parsing the trace file serially, in order to measure the memory accesses to the instruction memory (OPCODE accesses) and the memory accesses to the data memory (DATA accesses). The profiler calculated also the dissipated energy (base + interinstruction energy) within the processor core. Finally, with the use of an appropriate memory simulator (provided by an industrial vendor), the energy consumed in the data and instruction memories was measured.

As test vehicle, three algorithms from matrix algebra have been examined, which all perform matrix-matrix multiplication employing (the corresponding programs in C are shown in Table 1) :

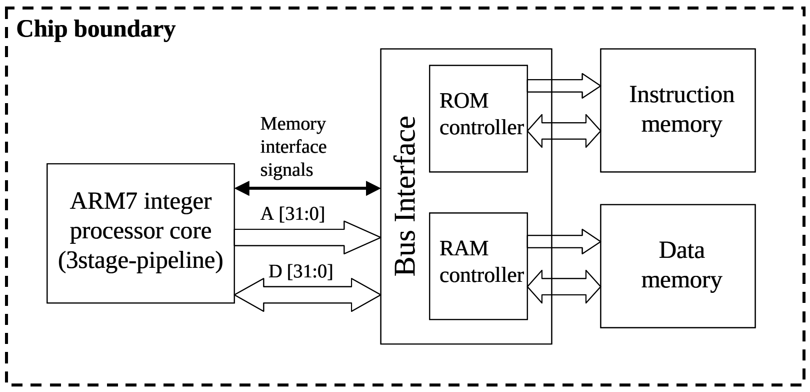

Fig. 1. Target Architecture.

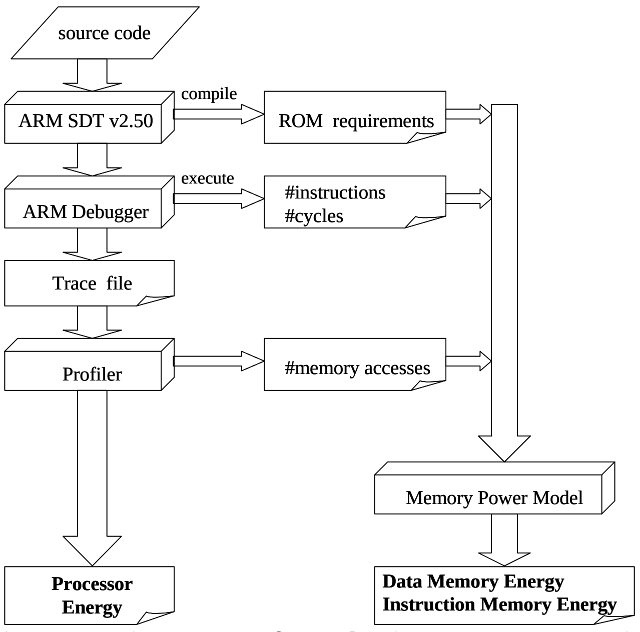

Fig. 2. Experiment set up for evaluating energy consumption.

Table 1. Matrix-matrix multiplication algorithms.

mmdot.c	mmsax.c	mmout.c
i=0;	k=0;	k=0;
while (i<m) {	while (k<n) {	while (k<p) {
j=0;	j=0;	i=0;
while (j<n) {	while (j<m) {	while (i<m) {
c=0;	i=0;	j=0;
k=0;	while (i<p) {	while (j<n) {
while (k<p) {	(&C[0][k]+j*n)=	(&C[0][0]+i*n+j)=
c=c+	*(&A[0][0]+i+j*p)*	*(&A[0][k]+i*p)*
(*(&A[i][0]+k))*	(*(&B[0][k]+i*n))+	(*(&B[k][0]+j))+
(*(&B[0][j]+k*n));	(&C[0][k]+j*n);	(&C[0][0]+i*n+j);
k++;	i++;	j++;
}	}	}
C[i][j]=c;	j++;	i++;
j++;	}	}
}	k++;	k++;
i++;	}	}
}		

a) a dot product computation (mmdot.c); b) a generalized SAXPY operation(mmsax.c); and c) an outer product update (mmout.c) [10]. The matrix multiplication is mathematically formulated as:

$$C = AB \quad \left(\Re^{m\times p} \times \Re^{p\times n} \rightarrow \Re^{m\times n}\right)$$

The second and third algorithm had no difference from our point of view on energy consumption and therefore we concentrated on the first two algorithms.

Next, we calculated the energy complexity of the algorithms. In the first one, there are two data accesses in the inner loop (elements of the matrices A and B) and one data access in the second loop (accessing elements of the matrix C, which contains the desired result of the multiplication). Thus the energy complexity associated with the data memory is calculated as:

$$E_{data\_mem1} = \left(2\cdot p\cdot n\cdot m + n\cdot m\right)$$

In the second source code, it is the inner loop that has four data accesses (the resulting matrix is accessed two times, whereas each of the matrices containing the initial numbers is accessed once). As a result, the energy complexity of the generalized SAXPY operation is calculated to be

$$E_{data\_mem2} = 4\cdot p\cdot n\cdot m$$

The computational complexity for the first algorithm (reflecting the energy complexity of the processor and the instruction memory as already described) was

$$E_{proc1} = E_{instr\_mem1} = \left(8\cdot p\cdot n\cdot m + 8\cdot n\cdot m + 9\cdot m\right)$$

while for the second one it was

$$E_{proc2} = E_{instr\_mem2} = \left(12\cdot p\cdot n\cdot m + 6\cdot n\cdot m + 9\cdot m\right)$$

In Table 2, the actual energy consumption for all system components (processor, instruction and data memory) is presented, based on the power models that have been developed, and using data from the ARM processor simulator. We can infer that although the number of executed assembly instructions has increased (by 28.5%) from the first algorithm (dot product) to the second (SAXPY), the energy consumed by the data memory has disproportionally increased (by 71.5%). Thus, considering only the computational complexity of an algorithm is not a safe method for estimating the approximate extent to which the data memory accesses will reach and as a result an inaccurate estimate of the dissipated energy might be obtained. On the other hand, energy complexity as proposed in this paper, complementary to the computational complexity can offer an improved way of estimating the energy consumption. In our example, the calculated doubling of data memory energy complexity from the dot product computation to the generalized SAXPY operation (namely 100% increase), is much closer to the observed increase of 71.5% in the data memory energy consumed and data memory accesses than the increase predicted by the computational complexity.

Fig. 3 illustrates the simulated and calculated results, when the complete energy complexity polynomial is considered, taking into account all three system components. The diagram shows the number of instructions, the simulated and estimated energy consumption of the second algorithm, normalized over the corresponding values of the first algorithm and thus the accuracy by which energy is estimated. It becomes obvious that energy complexity is a substantially better indicator than the computational complexity, regarding energy. Such an improvement in accuracy becomes extremely important as the size of the data memory (and therefore the corresponding energy) become larger. That is because in case of larger data memory sizes, the contribution of the data memory to the overall energy consumption becomes crucial.

Table 2. Energy consumption of matrix multiplication algorithms (mJ).

3x3	mmdot.c	mmsax.c	mmout.c
Cycles	564	726	750
Executed Assembly Instr.	410	527	561
Opcode Mem. Accesses	410	527	561
Data Mem. Accesses	63	108	108
Processor Energy	0.000723525 mJ	0.000933756 mJ	0.000964283 mJ
Instr_mem. Energy	0.001472 mJ	0.001768 mJ	0.001848 mJ
Data_mem. Energy	0.000859 mJ	0.001473 mJ	0.001474 mJ
Total Energy	0.003054525 mJ	0.004174756 mJ	0.004286283 mJ

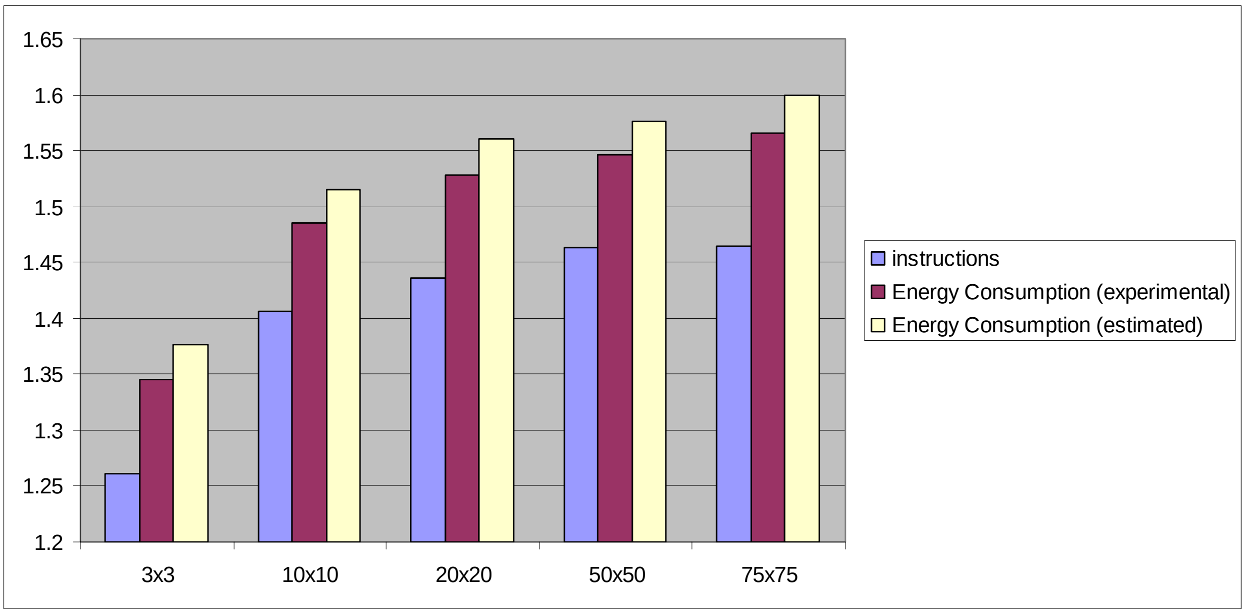

Fig. 3. Comparison of cycles, experimental energy consumption and estimated energy consumption (normalized values)

5. Conclusions and Future Work

Energy consumption is nowadays a major concern for the development of portable computing devices and it has been shown that software affects the energy dissipation to a large extent. In this paper, we have introduced the notion of energy complexity of an algorithm as a means to estimate the effect of data and instruction memory accesses to the total power consumption of an algorithm. An example of matrix multiplication algorithms has demonstrated the improvement that can be gained by considering energy complexity polynomials for the corresponding algorithms.

Future research on this field should include experimentation on larger and more complicated programs to establish the accuracy of the energy complexity as an estimate for energy consumption. Furthermore, the effect on energy of cache memories, possibly with multiple levels, as well as that of other architectural features should be studied.

References:

[1] V. Tiwari, S. Malik, A. Wolfe, Power analysis of embedded software: A first step towards software power minimization, *IEEE Transactions on VLSI Systems, Vol. 2* (1994), pp. 437-445.

[2] A. Chandrakasan, R. Brodersen, *Low power digital CMOS design* (Kluwer Academic Publishers, Boston, 1995).

[3] J.L.Henessy, D.A.Patterson, *Computer architecture: A quantitative approach* (Morgan Kaufman, San Mateo, CA, 1990).

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to algorithms*, 2nd edn. (MIT Press 2001).

[5] P. Landman, *Low-power architectural design methodologies*, Doctoral Dissertation (U.C., Berkeley, 1994).

[6] N. Kavvadias, P. Neofotistos, S. Nikolaidis, K. Kosmatopoulos, T. Laopoulos, Measurements analysis of the software-related power consumption of microprocessors, *IEEE Transactions on Instrumentation and Measurement, vol. 53*, no 4, pp.1106-1112, August 2004.

[7]http://www.arm.com/armtech/ARM7TDMI?OpenDocument, ARM Products and Solutions.

[8] S. Furber, *ARM system-on-chip architecture* (Addison-Wesley, Harlow, UK, 2000).

[9] A. Chatzigeorgiou, D. Andreou, S. Nikolaidis, Description of the software power estimation framework, IST-2000-30093/EASY Project, Deliverable 24, February 2003, URL: http://electronics.physics.auth.gr/easy.

[10] G.H.Golub, C.F.van Loan, *Matrix computations* (John Hopkins University Press, Baltimore, 1996).