ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

An Efficient Instruction-Level Energy Estimation Model for Embedded Systems

Devi. K. S*, Prof. V. Gopi

* PSN College of Engineering & Technology, Tirunelveli, Tamil Nadu, India

deviksudhir@gmail.com

Abstract

To optimize the energy consumption embedded systems, the estimation of energy consumption of the embedded applications are very important. This paper proposes a simple but effective instruction-level energy estimation model for embedded systems. For case study purposes, the model parameters were determined for a commonly used ARM9TDMI-based microcontroller. The total energy consists of the energy consumed by the processor core, flash memory, memory controller, SRAM etc. The model parameters that are determined includes op-code of instructions, number of shift operations, register bank bit flips, instructions weight and their Hamming distance, different types of memory accesses, the effect of pipeline stalls etc. To validate the proposed model, a physical hardware platform was developed which is having energy measurement capabilities. For several experiments conducted on various embedded applications from MiBench benchmark suite and less than 6% error in the energy consumption estimation was shown. Also an energy profiler tool was developed for the systems that use ARM9TDMI processors which provides valuable information and guidelines for software energy optimization.

Keywords: Embedded System.

Introduction

Embedded systems play a major role inday to day life of people in different areas. Mobile phones, washing machines, satellites etc are a few examples for devices that are having a processor embedded on them. A wide group of systems are mobile, battery powered devices, with limited source of energy. Thus, consumption of energy is an important aspect in embedded system design phase- for any application, itself. This helps the designers in optimizing the battery lifetime. Software is responsible for the large portion of energy consumption, in the case of an embedded system. Hence, an efficient model is necessary for the energy estimation. Mainly, there are two models for embedded instruction level energy optimization: measurement-based and simulation-based.

In the simulation-based approach a simulation model of the target hardware is used to run the applications and calculate the energy consumption of each part of the system which may be as detailed as gate level[2] or as abstract as behavioral level[3]. This approach needs the simulation model of all hardware modules that are mostly unavailable or very expensive. Also, evaluating the impact of a small change in an instruction opcode or operands on the total energy consumption of the system requires rerunning the simulation.

Measurement-based methods use data obtained from a physical target device. Most of the models [4],[5],[6], associate the instructions with the corresponding energy cost. The total application energy consumption is the aggregate cost of all executed instructions that can be calculated by running the application in an emulator. The main advantage of measurement-based methods is high accuracy in the energy estimation due to the real values obtained from the target platform.

It should be noted that for most of the commercially used microcontrollers (e.g., AT91SAM7X256 considered in this paper) there is no authentic SPICE (or any other detailed) sim- ulation model; hence, it is not possible to adopt a simulation- based approach. Furthermore, we believe that as measurement- based approaches use physical and real systems, they are al- ways more accurate than simulation-based approaches. Hence, in this paper we adopt a measurement-based approach.

Energy estimation models are also categorized according to their scope. The models presented in [4], [5], and [7] only model the processor core and the model presented in [6] models an embedded system including a microcontroller, external RAM, and external A/D converter.

In this paper, we introduce a new instruction-level energy estimation model and tool

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

for an ARM9TDMI-based micro- controller including the processor core, internal Flash and SRAM memories. While our model is simple enough to be implemented in any instruction set simulator, it provides good accuracy and can be used to estimate the energy consump- tion of the processor core (ARM9TDMI), SRAM, and Flash memory units of an embedded system. Also, the process of deriving the model allows for easy recalibration of the model for other platforms. The model is based on parameters such as instructions type, number of executed shift operations, register bank bit flips, weight and Hamming distance of the instruction words, and it has been validated using the MiBench benchmark suite [8]. Also, different types of memory accesses and pipeline stalls have been considered.

Compared with previous models, the new model provides;

1) equal or better accuracy as compared to [3], [7], and [9]. But this comparison might not be applicable to other previous works.

2) does not need cycle-accurate simulation that improves the simulation speed (unlike the model [4]).

3) is validated by a physical hardware implementation using MiBench [8] benchmarks that are considered as representatives for embedded applications.

4) proposes a simpler model for estimating the inter in- struction energy consumption that improves the simulation speed of large workloads.

5) is suitable for a wide range of applications (unlike application specific estimation models such as the model[6], [12]).

6) considers the energy consumption of Flash memory (unlike [4], [5], [7], [9], [11], and [6]) and SRAM (unlike [4], [5], [7], and [9]). Those previous works that consider memory units usually report the energy consumption of memory units as a whole. However, here, the energy consumptions of SRAM and FLASH units have been considered separately, enabling designers to analyze and optimize the energy consumption of each unit individually.

Most of the studies calculate the model coefficients by running sequences of instructions and measuring the energy consumption of the system during the execution cycles of each instruction. But certain studies[7] have used a "black-box" or "stimulus-response" approach where a set of test programs are executed on the hardware platform, the energy consump- tion is measured, and the model parameters are extracted using the regression methods. As this approach does not require detailed information about the internal structure of the processor, it improves the model retargetability. This model is similar to [4] and it is tested by executing randomly generated instructions on an ARM7TDMI core. It only estimates the energy consumption of the processor and does not cover the energy consumption of memories or peripherals.

Here, a new simplified version of the processor energy estimation model has been introduced. The parameters of this simplified model do not require cycle-accurate simulation. The energy consumption of memories is also modelled [11] but with a smaller number of parameters. For hardware energy measurement a stimulus-response approach similar to [7] has been used.

II. Measurement Method and System Architecture

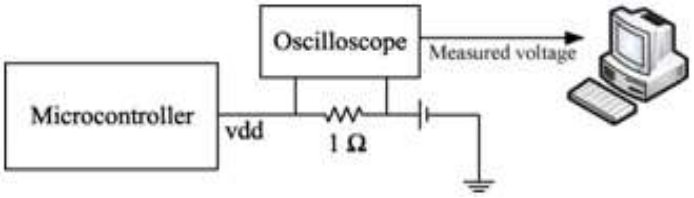

Fig. 1. Architecture of our measurement system

Precise measurement of the energy is one of the most important challenges in utilizing measurement-based methods. Some measurement-based studies [13],[1],[14] calculate the energy consumption by reading the average current drawn by the system from the power supply. We have used a similar approach by placing a 1-ohm resistor at the power supply pin of the microcontroller. The measured current (which is read by a high frequency oscilloscope) and the application execution time are sent to the host computer for calculating the energy consumption. Fig. 1 illustrates this architecture.

Here, an AT91SAM7X256 microcontroller [15] (based on ARM9TDMI processor core) is used as the target platform. The internal structure of this microcontroller is shown in Fig. 2. As we only concentrate on the energy consumption of the processor core, SRAM and Flash, all other modules are disabled by setting the appropriate flags in the initialization part of the program source code. The Flash memory is used for storing the code and read-only data while the SRAM is utilized as runtime data memory. The ARM9TDMI core is a 32-bit RISC microprocessor specialized for low-power applications and it is capable of achieving very high MIPS per watt with a five-stage pipeline and cache memory. The pipeline stages include instruction fetch (IF), decode (ID), and execute (EX), memory (M) and write back(WR). ARM9TDMI is a successor to the popular

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

ARM7TDMI core. ARM9 supports both 32-bit ARM and 16-bit Thumb instruction sets.

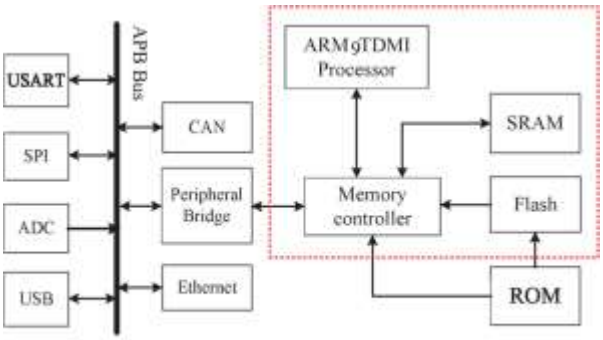

Fig. 2. Microcontroller internal structure. Our proposed model estimates the energy consumption of the CPU, Flash memory, SRAM, and memory controller.

Proposed energy consumption model

Most software programs that run on embedded processors consist of two parts.

1) An initialization part that configures system modules, initializes program variables, etc. This part of the pro- gram is executed only once at the start of the program in order that the system gets ready to perform its main operation.

2) The main part that is usually implemented as an endless loop.

From an energy consumption viewpoint, we can ignore the initialization part and assume that the system always operates in its main part. This is because when we turn an embedded system on, it is in the initialization phase for only few micro seconds and then it goes into the main phase where it operates for hours. This implies that almost all the energy consumption of an embedded system is because of the main phase and not the initialization phase. Based on this assumption, in this paper, we do not consider applications where there is no differentiation between initialization and main parts.

The total energy consumption of each instruction is expressed as the sum of fetch energy (E_{fetch}), decode energy (E_{decode}), execute energy ($E_{execute}$),memory state (Ememory), write back (Ewrite) and static energy (E_{static}), where

$$E_{fetch} = E_{cntr(code)} + E_{Flash(code)} + E_{IF}$$

$$E_{decode} = E_{ID}$$

$$E_{execute} = E_{EX} + E_{cntr(data)} + E_{Flash(data)} + E_{SRAM} + E_{stall} \quad (1)$$

($E_{\text{cntr(code)}}$ is the amount of energy consumed in the memory controller caused by the code, $E_{\text{cntr(data)}}$ is the amount of energy consumed in the memory controller caused by the data, $E_{\text{Flash(code)}}$ is the amount of energy consumed in the Flash memory caused by the code and $E_{\text{Flash(data)}}$ is the amount of energy consumed in the Flash memory caused by the data.) Since we have disabled all modules except the core, memories and the memory controller, there must be no instruction that causes energy consumption in the peripherals. However, the microcontroller datasheet does not clearly specify how the peripherals are disabled. If their power supply lines are completely cut off, then there will be no E_{static}. But if the supply lines are not cut off, and hence, the circuitry is turned on but inactive, they may consume a small amount of static power, which means E_{static} will not be zero.

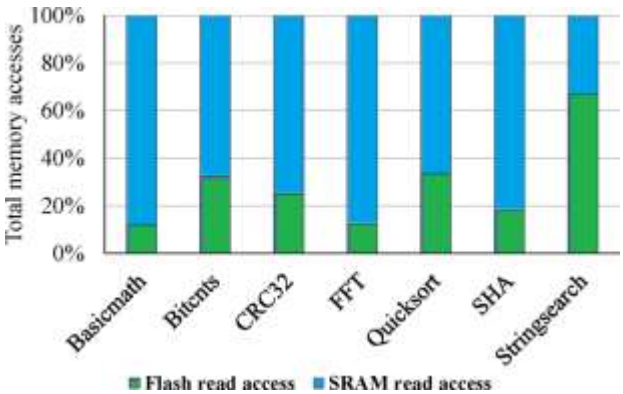

Fig. 3. Read access count of the Flash memory and SRAM in the MiBench benchmark suite applications

E_{memory} only depends on the number of memory accesses (for data transfer) during the program execution. Note that cal- culating some of these energy segments needs cycle-accurate simulation of the program, however, we estimate them using other parameters that can be calculated during instruction-level simulations.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

A. Memory Energy Consumption Model

Generally, the energy cost of accessing the RAM can be modeled with the number of read and written bytes. Analyzing the MiBench benchmark suite[8] applications (Fig. 3) shows that about 30% of the memory accesses target the Flash memory. To analyze the difference between the energy consumption of the Flash memory and SRAM, another experiment was conducted with two sets of test programs (Benchmark 1 and Benchmark 2). These benchmarks are iden- tical, except for the target address space. While the target ad- dress space of Benchmark 1 is within the Flash (0×100 000), the target address space of Benchmark 2 is within the SRAM (0×200 000). These benchmarks were executed with different target addresses and their energy consumptions were mea- sured. We observed that the average energy consumption of Benchmark 1 is 20% higher than that of Benchmark 2. The results also show that the energy consumption of accesses to different locations of the same memory module (SRAM or Flash) are almost constant. Hence, only the number and type of the memory accesses are considered in the memory energy estimation model and not the target address. Due to the notable share of the Flash memory access in the total application memory accesses (see Fig. 3) and the difference between the energy consumption of Flash memory accesses and SRAM accesses, two separate energy parameters are considered for the Flash and SRAM read accesses in our energy model. In the context of embedded systems, Flash write operations are relatively rare and are usually performed during the offline phase (programming/configuration phase) of an embedded system where system can be connected to elec- tricity power lines and does not use battery. However, during the normal operation of embedded systems where system is battery-operated (and hence energy consumption and energy estimation is prominent), the system usually accesses the Flash by read operations (for example to fetch instructions), and hence, we have to consider Flash read operations in energy estimation.

B. Processor Core Energy Consumption Model

The energy consumption of the different pipeline stages are grouped together as the processor core energy consumption. Some previous studies have shown that the energy consumption of the processor core during the execution of an instruction can be divided into three parts: *i)* the base energy cost, which only depends on the current instruction, *ii)* the interinstruction cost, which is the amount of energy consumed by the processor during the consecutive execution of different instructions, and *iii)* the pipeline stall cost [4], [5], [7], [9]

In most cases, the interinstruction energy cost is about 5% of the base instruction cost [9]. Thus, we simplify the model by ignoring detailed interinstruction cost estimations and use other parameters, considering the internal structure of the processor core [16]. These parameters include the Hamming distance and weight of the instructions, number of bit flips in the register bank, and the number of shift operations.

1) Instructions Word Hamming Distance*:* Any change in the input signals of a circuit can cause a series of activities in the circuit. In some designs these subsequent activities and their corresponding energy consumption have a direct relation- ship to the changes in the input signals. To exploit this idea in our proposed model, the effect of the instruction Hamming distance on the core power consumption was analyzed by running a series of benchmarks. Each of these benchmarks contains a large amount of two different types of instruc- tions. Benchmark 1 comprises two separate blocks where all instructions of each block are of the same type. Benchmark 2 orders the instructions in an interleaved manner where each instruction of type A is followed by an instruction of type B and vice versa. The energy consumption of Benchmark 2 shows about 6% increase over that of Benchmark 1. We model this portion of energy consumption by simply counting the Hamming distance between the instructions during the execution flow

2) Instructions Word Weight: The energy consumption of an instruction also depends on the operands' weight (number of "1") which is the result of using dynamic CMOS logic in the design of ARM9TDMI [11]. According to our exper- iments, we can consider a single weight energy coefficient for all instructions that will cause a negligible error in energy estimation but it will make the model simpler. Therefore, the instruction weight is added as a parameter to our model.

3) Number of Shift Operation*:* To observe the possible effects of the shift operation on the energy consumption of the system, we conducted two different sets of experiments. One group consists of a large number of instructions with shifted operands while the other one contains the same instructions but without using this option. The experiments were repeated for all instructions that can use the shifted operand option and the results show that the shifted version consumes up to 28% more energy than normal version.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

4) Register Bank Bit Flips*:* A number of benchmarks based on MVN instruction were prepared to observe the effect of the register bank activity on the system energy consumption. The MVN instruction reads the value of the source operand, performs a bitwise NOT operation and then assigns the result to the destination register. Table I lists the benchmarks used in this experiment.

TABLE 1- Register Bank Benchmarks

No:	Benchmark 1	Benchmark 2	Benchmark 3
1	MVN R1,R2	MVN R1,R1	MVN R1,R2
2	MVN R1,R2	MVN R1,R1	MVN R1,R2
..	..	..	..
10000	MVN R1,R2	MVN R1,R1	MVN R1,R2

In all benchmarks, there will be no further activity on the instruction bus or data bus after executing the last MOV instruction. In Benchmark 1, only the first MVN command results in register bank activity and the consequent instructions write the same 0×FFFFFFFF value into R1 register. Benchmark 2 uses the same register for input and output and causes 32 bit flips per cycle in R1 register. Energy measurement of Benchmark 2 shows 3% increase over that of Benchmark 1. To clarify whether the energy consumption increase is related to the Hamming distance or the weight of the registers, we conducted another experiment using Benchmark 3. Benchmark 3 initializes all the registers to value 0×FFFFFFFF in order to increase the total register bank weight to the maximum possible value. Benchmark 1 on the other hand, initializes all the registers to 0 that lowers the total weight of the register bank to the minimum possible value. The energy measurement result of Benchmark 3 shows about 0.1% deviation from the energy consumption of Benchmark 1 that indicates that the register bank energy consumption is more related to the Hamming distance than the weight. Considering the outcome of this experiment, the number of bit flips in the register bank was added as a new parameter to the energy estimation model.

5) Pipeline Stall*:* So far, all parts of (6) are covered with the exception of E_{stall} . There are some instructions that can cause a pipeline stall that will lead to additional energy consumption in the system. These instructions include some types of memory access, multicycle instructions (such as MUL), and the wrong branch prediction. Hence, only the EX stage of the pipeline is active and the other stages are stalled and consume a constant amount of energy.

According to their duration, pipeline stalls can be grouped as follows.

a) The fixed length pipeline stall that delays the execution for a constant amount of cycles. (e.g., the SWP instruc- tion that takes four cycles to execute). The extra energy consumption of these pipeline stalls is added to the base energy cost of the corresponding instructions.

b) The variable length pipeline stall whose duration de- pends on some items like operand values, previously accessed memory addresses and correct prediction of target of Jump instructions.

Estimating the exact energy consumption of the variable length pipeline stalls needs cycle-accurate simulation of the program. We analyzed the MiBench benchmark suite [8] applications and chose the average energy consumption of the each multicycle instruction as the base energy cost of that instruction. Although this approximation method causes a small error in the energy estimation model, it eliminates the need for cycle-accurate simulation.

Analysis

To compute the coefficients of the model, a statistical method known as regression analysis [15] is used. The energy model is assumed to be a linear polynomial function and the factors are extracted using linear regression analysis. In the linear regression analysis the model is in the following form [15]

$$y = c0 + c1\,x1 + c2\,x2 + \ldots + c_n\,x_n + e$$

where c_i is the regression coefficient , x_i is the data vector, y is the dependant value and e is the error of approximation. To find the c_i coefficients the least square method [15] was used. Given M independent equations, regression analysis tries to extract the unknown coefficients by assigning different values to them, re-evaluating the polynomials and comparing the result of each polynomial with the corresponding dependant value y. This procedure, known as fitting [15], is repeated for all available equations until either convergence of all the coefficient values or reaching the maximum iteration.To provide the required

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

equations for the fitting procedure,60 special test programs were prepared and their energy consumption were measured. Each test program was designed to magnify the effect of one specific model parameter. This improves the accuracy of the fitting procedure by reducing the regression model's complexity. The combination of energy measurement values of these benchmarks and their profile report (analyzed with Sim-profile), form 60 equations for the fitting procedure. The ARM9TDMI datasheet indicates that most instructions have two different forms that only one of them updates the flag register. For example ADD and ADDS both perform the addition operation but only ADDS updates the flag registers (e.g., Z flag is set when the result is zero or V flag when an overflow occurs). In the first version of the model, one coefficient was assigned for each one of these pairs. But in order to achieve more accurate results without imposing unnecessary complexity on the model, for some instruction pairs separate coefficients is assigned for each member of the pair. In order to find the most effective instructions in total application energy consumption, the MiBench benchmark suite applications were analyzed with the proposed energy model. The top 10 contributors are listed in Fig. 4. According to the results, the two versions of ADD and MOV instructions were separated, which improved the estimation accuracy of the model and confirmed the validity of our decision.

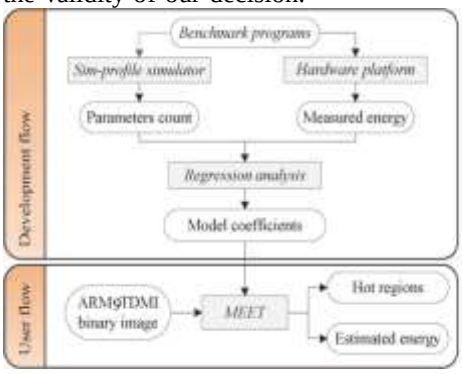

Fig. 5. Energy estimation model derivation process.

Fig. 5 summarizes the approach that we have followed in this paper to develop the energy estimation model. It also illustrates the flow that the user of our tool must follow to estimate the energy consumption of the application and identify the hotspots of the source code.

TABLE II Final Results of Regression

Instruction parameters	
Parameter	Energy(nJ)
ADD	0.89
RSB	1.153
BIC	1.049
MVN	1.13
AND	1.173
CMP	0.978
ADC	1.127
LDR	1.84
ORR	1.131
B	0.79
CMN	0.976
SUB	1.143
MOV	1.284
SBC	1.113

Results

The final results of the regression analysis is summarized in Table II. To measure the quality of the regression we used a criterion named coefficient of determination, which is denoted by R^2 [15]. The reported coefficient of determination for our final regression analysis is 0.9987, which means 99.87% of all variations of the test applications energy consumption are captured by the proposed energy estimation model.

In order to test the accuracy of the model in real world applications, a set of experiments were conducted on MiBench benchmark suite applications. The energy consumption of all applications are estimated with less than6% error. Some sources of error in our energy estimation model are ignoring the Hamming distance and weight of address bus and data bus and using a constant value to model the energy consumption of variable length pipeline stalls to avoid cycle-accurate simulation of the system. Considering the value of R^2 it is expected to achieve better accuracy, which means the test programs can be improved to cover all aspects of microcontroller energy consumption. Having the final regression results, the energy coefficients were embedded in the Sim-profile simulator from Sim- pleScalar toolset to make the process of energy estimation easier. Since Sim-profile has the ability to profile

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

the applica- tion based on a given simulation parameter, adding the energy consumption estimation ability makes it an energy profiler tool. This profiler tool is called MEET. MEET can be used for finding blocks of software with high energy consumption. It can also provide useful guidelines about optimizing the energy consumption by reporting the cause of high energy consumption in hot regions.

Conclusion

An instruction-level energy estimation model for a microcontroller-based embedded system was presented. By combining similar energy coefficient values, the presented model became much simpler than most proposed models that can speed up the energy estimation process and therefore the development of the embedded systems. The model was validated against a physical hardware platform, and the error of estimation for a number of applications from the MiBench benchmark suite was less than 6%. Retargetability is another advantage of our proposed model that makes it easy to adjust the model coefficients for a new platform. The model coefficients can be calibrated by measuring the energy consumption of test programs for the new platform and rerunning the regression analysis.

Also, a tool called MEET was developed, which receives a binary ARM9 application and profiles the energy consumption of the application on the microcontroller. The results can identify the hotspots of the code and help in selecting the best coding technique in order to optimize the total energy consumption of the application.

Reference

1. *V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," IEEE Trans. VLSI Systems, vol. 2, no. 4, pp. 437–445, Dec. 1994.*
2. *B. Klass, D. E. Thomas, H. Schmit, and D. F. Nagle, "Modeling inter- instruction energy effects in a digital signal processor," in Proc. Digital Signal Processor, Power-Driven Microarch. Workshop in Conjunction with Int. Symp. Comput. Arch., Jun. 1998, pp. 18–23.*
3. *G. Callou, P. Maciel, E. Tavares, E. Andrade, B. Nogueira, C. Araujo, and P. Cunha, "Energy consumption and execution time estimation of embedded system applications," Microprocessors Microsyst., vol. 35, no. 4, pp. 426–440, 2011.*
4. *N. Chang, K. Kim, and H. G. Lee, "Cycle-accurate energy consumption measurement and analysis: Case study of ARM7TDMI," in Proc. ISLPED, 2000, pp. 185–190.*
5. *S. Nikolaidis, N. Kavvadias, T. Laopoulos, L. Bisdounis, and S. Blionas, "Instruction level energy modeling for pipelined processors," J. Embed- ding Comput., vol. 1, no. 3, pp. 317–324, Aug. 2005.*
6. *V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, and T. Laopoulos, "Energy consumption estimation in embedded systems," IEEE Trans. Instrum. Meas., vol. 57, no. 4, pp. 797–804, Apr. 2008.*
7. *S. Lee, A. Ermedahl, and S. L. Min, "An accurate instruction-level energy consumption model for embedded RISC processors.*
8. *M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in Proc. IEEE Int. Workshop Workload Characterization, Dec. 2001, pp. 3–14.*
9. *N. Kavvadias, P. Neofotistos, S. Nikolaidis, K. Kosmatopoulos, and T. Laopoulos, "Measurements analysis of the software-related power consumption of microprocessors," IEEE Trans. Instrum. Measurement, vol. 53, no. 4, pp. 1106–1112, Aug. 2004.*
10. *K. Zotos, A. Litke, E. Chatzigeorgiou, S. Nikolaidis, and G. Stephanides, "Energy complexity of software in embedded systems," in Proc. IASTED, Jun. 2005, pp. 17–27.*
11. *S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel, "An accurate and fine grain instruction-level energy model supporting software opti- mizations," in Proc. Int. Workshop PATMOS, Sep. 2001.*
12. *V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, and T. Laopoulos, "Energy consumption estimation in embedded systems," in Proc. IEEE IMTC, Apr. 2006, pp. 235–238.*
13. *M. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," IEEE Trans. VLSI Syst., vol. 5, no. 1, pp. 123–135, Mar. 1997.*
14. *J. T. Russell and M. F. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors,"*

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

in Proc. ICCD: VLSI Comput. Processors, Oct. 1998, pp. 328–333.

15. *S. Chatterjee and A. S. Hadi, Regression Analysis by Example. 4th ed., New York, USA: Wiley, 2006.*