An Accurate Instruction-Level Energy Consumption Model for Embedded RISC Processors

Sheayun Lee*
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
sylee@archi.snu.ac.kr

Andreas Ermedahl†
Dept. of Information Technology
Uppsala University
SE-751 05 Uppsala, Sweden
ebbe@docs.uu.se

Sang Lyul Min*
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
symin@dandelion.snu.ac.kr

ABSTRACT

Energy consumption of software is becoming an increasingly important issue in designing mobile embedded systems where batteries are used as the main power source. As a consequence, recently, a number of promising techniques have been proposed to optimize software for reduced energy consumption. Such low-power software techniques require an energy consumption model that can be used to estimate or predict the energy consumed by software. We propose a technique to derive an accurate energy consumption model at the instruction level, combining an empirical method and a statistical analysis technique. The result of the proposed approach is given by a model equation that characterizes energy behavior of software based on the properties of the instructions. Experimental results show that the model equation can accurately estimate the energy consumption of random instruction sequences, with an average error of 2.5 %.

Keywords

Low-power systems, instruction-level energy model, regression analysis

1. INTRODUCTION

Energy consumption of software has recently emerged as an important metric of system performance with the increasing requirement for low-energy computing. Especially for embedded systems, there is a high demand for optimization techniques that enable energy reduction for software, since an increasing number of applications are powered by batteries. Therefore, recent studies have been focusing on developing techniques to reduce the energy consumption at various levels, including program optimization for low power [10, 15, 20]. Such low-power program optimization techniques require a detailed cost model represented in terms of energy consumption to direct the decisions on program transformations.

We propose a technique to build an accurate energy consumption model at the *instruction level*, which can be used in estimating the energy consumption by a given sequence of instructions. For the energy consumption model to be applicable to a program optimization framework, it must have the following properties.

- **Accuracy**: The model should be able to estimate the energy consumption accurately.
- **Simplicity**: The model should be constructed using simple properties visible at the instruction level, so that it can be easily integrated into a program optimizer.
- **Accountability**: The model should be able to identify the significance of each factor that affects the energy consumption, so that it can direct the optimization.
- **Retargetability**: The model building framework should be general enough to be applied to a wide range of different target processors.

To construct the energy consumption model, we first assume a model equation which is a linear combination of the factors that can possibly affect the energy behavior of instructions. Then we derive the unknown parameters of this equation using a "black-box" or "stimulus-response" approach, where we apply a set of test programs (stimulus) and observe the energy consumption measured from the real hardware (response). For this purpose, a statistical analysis technique called *linear regression analysis* is used to fit the model equation to the actual energy behavior of the target processor.

The proposed approach gives a detailed instruction-level characterization of the energy behavior of processors, enabling a simple but yet accurate estimation of software energy consumption. We expect that this information will

* This work was supported in part by the Ministry of Education under the BK21 program, and by the Ministry of Science and Technology under the National Research Laboratory program.

† This work is performed within the Advanced Software Technology (ASTEC, `http://www.docs.uu.se/astec`) competence center, supported by the Swedish National Board for Industrial and Technical Development (NUTEK, `http://www.nutek.se`).

provide a base for aggressive program optimization for low power, which becomes increasingly important in embedded systems design. Furthermore, the availability of such a model is essential in developing a systematic optimization framework that considers different optimization criteria at the same time, such as code size, execution time, and energy consumption.

The major contributions of this paper are as follows. First, the technique provides a systematic approach for estimating software energy consumption based on a simple linear model equation. Second, the resulting energy model can be used to identify important factors that have significance in the energy consumption by instructions. Finally, the proposed model building approach is generally applicable to a wide variety of processors, since it is based on an empirical method that does not require information about the processor implementation.

In this paper, we focus on estimating the energy consumed inside the core of a RISC-style microprocessor by characterizing the instructions executed on the processor. However, we believe that a similar approach can be applied to model the system-level energy behavior, by taking into account various components such as memory devices and peripherals.

The rest of the paper is organized as follows. Section 2 discusses previous works related to our research. In Section 3, we describe in detail the proposed approach for deriving an instruction-level energy consumption model. Our experimental setup and results are presented in Section 4. Finally, Section 5 concludes the paper by outlining possible future extensions to this work.

2. RELATED WORK

Recently, attempts have been made to construct energy consumption models for software. The previous approaches are based either on functional simulation of the processor, or on direct measurement of energy. 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. For example, Mehta et al. [14] propose a power profiler that records the information of the previous and the current states of functional units, as well as the correlated switching capacitance. As an extension to this approach, Chen et al. [6] present a technique that can estimate the cycle-level energy consumption data based on hierarchical decomposition of the architectural features of the target processor. A more generalized form of an RT (register transfer) level energy simulator called SimplePower is proposed in [22]. While the above approaches are concentrated on modeling the target architecture, Klass et al. [12] analyze the effect of sequential execution of different instructions, using a gate-level analysis tool. In their approach, the inter-instruction energy effect is modeled by additional energy consumption observed when each instruction is executed after a NOP instruction. A common drawback of these simulation-based energy models is that they do not provide a mechanism that can calculate the energy consumption of software directly from the instruction sequence.

On the other hand, in the *measurement-based* approaches, the energy consumption of software is characterized by examining the data obtained from real hardware. The advantage of the measurement-based approaches is that the resulting energy model is close to the actual energy behavior of the processor, because the data is acquired from the hardware itself. Tiwari et al. [19] describe a technique to model the energy cost of software, based on the average current drawn by the target processor. In this approach, the energy model is given by a power cost table that records the unique base cost for each instruction and the inter-instruction effects. The base cost for an instruction is defined as the average current drawn by this instruction executed repeatedly in a tight loop, multiplied by the number of cycles taken by each instance of the instruction. On the other hand, the inter-instruction effect is defined as the additional power cost incurred by executing different instructions sequentially. However, recording this inter-instruction effect significantly increases the size of the power cost table, which requires $O(N^2)$ space where N is the number of instructions in the instruction set. To rectify this problem, a technique to group the instructions into common classes [13] is proposed. These techniques provide a simple framework for software energy estimation by summarizing the energy consumption by instructions in the form of a table. However, by relying on the average current, they largely ignore the detailed impacts of various factors that affect the energy consumption at the instruction level. Moreover, these techniques do not provide the information about the energy variation due to various aspects of instructions such as the instruction fetch address and the operand specifiers.

In contrast to the above approaches based on the average current, Russell and Jacome [16] present a software energy estimation model based on instantaneous power measured by a digitizing oscilloscope. A technique to derive more fine-grained energy consumption is proposed by Chang et al. [4], where they measure the cycle-level energy consumption using a measurement hardware developed in the research. They also analyze the impact of various properties of instructions on the energy consumption, based on the measurement. Using this approach, it is shown that the energy consumption of software is dependent on the properties of instructions, such as register numbers, immediate operands, etc. However, their instruction-level energy characterization of software does not give a framework that can be used to estimate the energy consumption of a given sequence of instructions.

To explain the complex energy behavior of processors, statistical analysis techniques are employed in energy estimation of software. Gebotys et al. [8, 9] propose an energy estimation and optimization technique for VLIW processors, incorporating a statistical method for analyzing the functional unit usage patterns of instructions. This approach tries to predict the energy consumption of software using regression analysis. The prediction is used to minimize the energy consumption with respect to the average current drawn. Recent studies by Brandolese et al. [2] and Sami et al. [17] also present techniques to estimate the software energy consumption using a combination of functional decomposition and a statistical analysis technique. These approaches are focused on modeling the energy consumption in terms of the usage of various functional units, mainly targeted for

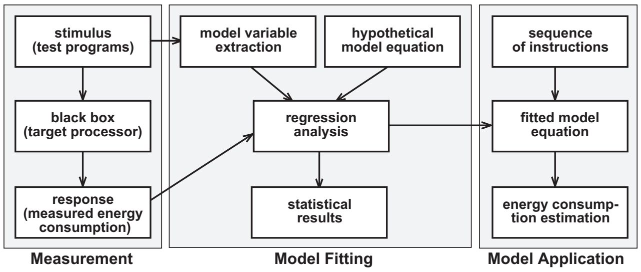

Figure 1: The overall approach for deriving an energy consumption model.

VLIW processors. The technique presented in this paper is distinguished from the above approaches in that we concentrate on the properties of instructions rather than on the functional units inside the processor.

In this paper, we aim at deriving an accurate energy consumption model at the instruction level, combining an empirical approach and a statistical analysis technique. By using an empirical method, we eliminate the need for detailed information about the target processor implementation. At the same time, the empirical approach provides retargetability of the model building framework, since it is not dependent on a specific target processor. By using a statistical analysis technique, we mathematically summarize the relationship among a number of factors that interact in a complex manner, each contributing to software energy consumption.

3. INSTRUCTION-LEVEL ENERGY MODEL

The energy consumed in a microprocessor is dependent on the internal organization and implementation of the processor. Therefore, one possible approach for estimating the energy consumption of software is to perform a circuit-level simulation of the target processor with the program as the input to the simulation. However, this simulation-based energy estimation is not practical for program optimization for reduced energy consumption for a number of reasons. For example, evaluating the impact of even a slight change in an instruction sequence requires rerunning the whole simulation process, which has substantial complexity. This makes the simulation-based method inapplicable to program optimization, which requires frequent cost evaluation of a number of optimization candidates. Moreover, the circuit-level implementation data for off-the-shelf microprocessors are not generally available, which makes the simulation-based approach more difficult to apply.

Therefore, to obtain an energy consumption model that can be incorporated into a program optimization framework, we propose a technique to derive an accurate energy consumption model at the instruction level. The proposed energy model can estimate the energy cost of software using the properties of instructions. Figure 1 shows our approach for deriving the instruction-level energy consumption model. Instead of relying on information about the processor implementation, we use an empirical approach, where the model is based on the actual energy consumption data measured from real hardware. That is, we regard the implementation of the target processor as a "black box" whose internals are not known, and assume that the only accessible information is the responses of this black box for a set of stimuli. In other words, we measure the energy consumption of the target processor when it executes a set of test programs, and try to derive an energy model by reasoning about the relationship between the instruction sequence and the measured energy. This empirical method significantly simplifies the process of model construction, since it does not require modeling of complex internal circuits of the target processor. Furthermore, the proposed method also enhances the retargetability of the approach, since the technique is not dependent on the implementation of a specific processor. Of course, the whole measurement procedure should be repeated when the energy consumption model is to be retargeted to a different processor. However, the model building framework itself is retargetable in the sense that the technique to derive the model equation can be applied to different processors without any major modification.

Note, however, that our approach is not confined to measurement-based construction of the energy model. That is, the technique can also derive the energy model using the data obtained from energy simulators that give information about cycle-level energy consumption, when the simulator for the target processor is already available. In this case, the proposed technique significantly reduces the time complexity of energy estimation, since once the model has been derived, we can estimate the energy consumption without the need for running the simulation for the instruction sequence under investigation. Also, the technique can provide the information about the various factors that affect the energy consumption of instructions in an abstract form of a model equation, which is impossible when we rely entirely on simulation in estimating the software energy consumption.

To derive the energy model by summarizing the data gathered from measurement or simulation, we use a statistical analysis technique called regression analysis [5]. First, we assume a hypothetical model equation with unknown parameters, which is a function of model variables defined in terms of various aspects of instructions. The model variables are the factors that may affect the energy consumption of instructions, whose values are extracted from the test programs by examining the instruction sequence. Then, the regression analysis determines the parameters of the model equation by investigating the combinations of the measured energy and the values of the model variables. This procedure is called model fitting [5], whose result is a fitted model equation that explains the relationship between the model variables and the energy behavior of the target processor. Using this fitted model equation, we can estimate the energy consumption of a given instruction sequence. In addition, the regression analysis will produce a set of statistical results, which can be used in refining the model as well as in measuring the quality of the fitted model equation.

In Section 3.1, we describe how the hypothetical model equation is constructed at the instruction level by introducing a set of energy formulas and model variables. In Section 3.2, we explain how we can derive the parameters of the model equation using linear regression analysis.

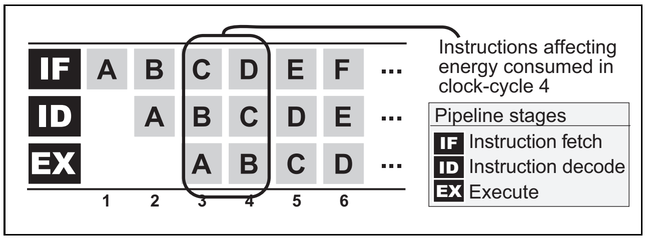

Figure 2: Example of pipelined execution scenario.

3.1 Energy Model Equation

In general, the energy consumption of a CMOS processor is dependent on the switching activity in a clock cycle as well as the current charge/discharge state of the circuit [4]. Specifically, the energy consumed in a clock cycle is proportional to (1) the number of bit flips in the internal signals in that clock cycle, and (2) the number of logical 1's (or the number of 0's, alternatively) in the signals. Since the internal signals in a processor are controlled by instruction execution, we conjecture that the energy consumption can be modeled by a combination of properties visible at the instruction level. This motivates us to define the model equation in terms of the currently executing instruction and the previously executed instruction at various stages in the execution pipeline.

More generally, for a pipelined processor, the energy consumed in a clock cycle is determined by the instructions that currently occupy each of the pipeline stages and those that previously occupied each of the stages. Assuming the energy consumed in a clock cycle is the sum of the energy consumed in all the pipeline stages, we calculate the energy consumption for a clock cycle by a simple equation. Let S be the set of all the pipeline stages and $I_s(i)$ the instruction occupying pipeline stage s in clock cycle i. Then the energy consumed at clock cycle i is given by

$$E_i = \sum_{s \in S} e_s(I_s(i), I_s(i-1)) \ , \tag{1}$$

where $e_s(X, Y)$ denotes the energy consumed in pipeline stage s when instruction X is executed in that stage, preceded by instruction Y.

For example, consider the pipelined execution scenario shown in Figure 2, where we have three stages: IF (instruction fetch), ID (instruction decode), and EX (execute). Assume that an instruction sequence of `A`, `B`, `C`, `D`, and `E` is executed sequentially. The energy consumed in clock cycle 4 is calculated by

$$E_4 = e_{\mathrm{IF}}(\mathtt{D}, \mathtt{C}) + e_{\mathrm{ID}}(\mathtt{C}, \mathtt{B}) + e_{\mathrm{EX}}(\mathtt{B}, \mathtt{A}) \ . \tag{2}$$

We model the energy consumed at pipeline stage s by instruction X executed after instruction Y, denoted by $e_s(X, Y)$, using the properties of instructions X and Y. Specifically, we define a set of model variables in terms of the binary representation of the factors that are assumed to influence the energy consumption. Examples of instruction-level model variables include:

- *instruction fetch address*, which can affect the energy consumption of the address bus when the instruction is fetched,
- *instruction bit encoding*, which can affect the energy consumption of the instruction register and the pipeline latches,
- *operand specifiers*, such as register numbers and the immediates, which can affect the energy consumption of instruction decoding and execution, and
- *data values*, which can affect the energy consumption of arithmetic and logical execution units.

Assume that we have identified all the model variables that have significance in the energy consumption. If we let V be the set of all the model variables, $e_s(X, Y)$ is given by

$$e_s(X, Y) = B_s^X + \sum_{v \in V} f_s^X(v_X, v_Y) \ , \tag{3}$$

where v_X and v_Y denote the model variable v defined by instructions X and Y, respectively. The term B_s^X gives the base cost for instruction X at pipeline stage s, which corresponds to the portion of the energy consumed by instruction X at stage s, regardless of the previous state of the pipeline. On the other hand, the function $f_s^X(v_X, v_Y)$ gives the variation of energy consumption according to the model variable v defined by the two instructions X and Y.

Based on the observation that the energy consumption is proportional to the number of bit flips and the number of logical 1's in the internal signals, we define the energy variation function as

$$f_s^X(v_X, v_Y) = H_s^{v/X} \cdot h(v_X, v_Y) + W_s^{v/X} \cdot w(v_X) \ , \tag{4}$$

where $h(i, j)$ denotes the Hamming distance between two binary numbers i and j, and $w(i)$ denotes the weight (number of 1's) of a binary number i. In Equation 4, $H_s^{v/X}$ and $W_s^{v/X}$ are unknown coefficients that characterize the energy consumption of instruction X at pipeline stage s, with regard to model variable v.

Note that we have a linear equation system for energy consumption in a clock cycle, with a number of unknown parameters, i.e., $H_s^{v/X}$'s, $W_s^{v/X}$'s and B_s^X's. We call these parameters *characterizing parameters*, since they describe the characteristics of the energy behavior of the target processor. In the next step of our proposed approach, the best estimates of these unknown parameters are determined by linear regression analysis.

It is possible that the number of such characterizing parameters is very large, depending on the number of pipeline stages and the number of different instructions. Moreover, the set of the characterizing parameters possibly includes those that are insignificant or even irrelevant to energy consumption, which possibly degrade the model accuracy. Therefore, to rectify this problem, the regression analysis uses two techniques based on statistical model testing, in addition to deriving the values of the characterizing parameters. First, we identify those parameters that have little or no significance and eliminate them, to reduce the number of characterizing parameters and enhance the model accuracy at the same time. Second, we merge those characterizing parameters for all the instructions or for a group of instructions, when the behavior of different instructions in the same group is similar with respect to one or more model variables. These techniques, together with the process of deriving the values of the characterizing parameters, are presented below.

3.2 Regression Analysis for Energy Model

Regression analysis is a statistical method for investigating functional relationships among variables [5]. The relationship is expressed in the form of an equation or a model connecting a *response variable* with one or more *predictor variables*. That is, when we denote the response variable by y and the set of predictor variables by $x_1, x_2, \ldots, x_p$, the true relationship between y and $x_1, x_2, \ldots, x_p$ can be approximated by a regression model

$$y = f(x_1, x_2, \ldots, x_p) + \varepsilon \ , \tag{5}$$

where ε is assumed to be an error representing the discrepancy in the approximation. In our case of deriving the energy model, the response corresponds to the energy consumption measured in each cycle, and the predictors to the Hamming distances and the weights of the model variables. [1]

When the relationship between the response and the set of predictors is assumed to be linear, the method for constructing the regression model is called *linear regression analysis*. That is, a linear regression model is expressed by an equation

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon \ , \tag{6}$$

where β_0, β_1, …, β_p are constants and called *regression coefficients*. The best estimates of these regression coefficients, i.e., the ones that lead to the model equation that best explains the relationship between the response and the predictors, are determined by investigating a number of sample combinations of the response and the predictors. The most common method used in finding the regression coefficients is called the *least square method* [5], which is the one that we use in our analysis.

In our case, finding the best estimates of the regression coefficients corresponds to determining the values of the characterizing parameters in our model equation. Compared to many other applications of statistical analysis, our case lends us the benefit of being able to freely set the values of the predictor variables, by using carefully written test programs. This means that we can determine a subset of characterizing parameters independently of the others, and use the values of these parameters later in the analyses for other parameters yet to be determined. For example, we can change the instruction fetch address while fixing all the other factors such as operand specifiers and all the data values, by repeatedly executing the same instructions. Then we can determine the impact of register numbers, for example, by executing the same instructions with different register numbers but with all the other factors fixed. Here, we calculate the value for the response variable by subtracting the variation due to the changes in the fetch address, which has been previously analyzed, from the measured energy value. Similarly, using the set of previously determined values of characterizing parameters, we can analyze the other parameters separately from the others. This stepwise derivation of characterizing parameters not only enhances the accuracy of the model by reducing the possible errors incurred in regression, but also keeps the complexity of the regression analysis low by decomposing the whole problem into several subproblems.

In addition to deriving the values of the characterizing parameters, we identify and remove those predictors that have little or no significance in the energy consumption, based on the statistical measures given by the regression analysis. This is done by testing a null hypothesis $H_0 : \beta_i = 0$ against the alternative $H_1 : \beta_i \neq 0$ for each regression coefficient β_i. The best model is selected using a model testing technique called the t-test [5]. Intuitively, to assess the significance of each predictor, we compare the model with a specific characterizing parameter set to zero with another model with the parameter set to the value derived from the regression analysis. Removing the predictors that are insignificant or even irrelevant to the energy consumption increases the accuracy of the resulting model equation, and also maintains the model complexity at a reasonable level.

In addition, we can reduce the complexity of the model equation further by merging specific characterizing parameters for all the instructions, or for a group of selected instructions. This is done by testing a null hypothesis $H_0 : \beta_i = \beta_j = \cdots = \beta_k$ against the alternative $H_1 : \beta_i \neq \beta_j \neq \cdots \neq \beta_k$ for a combination of selected regression coefficients β_i, β_j, …, β_k. The model that best explains our measured energy values is selected based on a model testing technique called the F-test [5]. Intuitively, to check if a group of instructions should be merged or not with respect to a specific characterizing parameter, we compare a model with the same characterizing parameters for different instructions and another model with different characterizing parameters for different instructions. For example, we naturally assume that the impacts of fetch address or register numbers are identical for all the instructions, while the impact of data values is different from one instruction to another. For our target processor, this assumption is validated using the technique explained above, in the regression analysis phase of the model derivation.

Besides the fitted model equation, the regression analysis also produces statistical measures that can be used to assess the validity of the model. The common measure of the quality of fit, i.e., the accuracy of the resulting model equation, is the *coefficient of determination* denoted by R^2 [5]. Intuitively, it is interpreted as the proportion of the total variability in the response variable that is accounted for by the set of predictor variables. In the rest of this paper, we will use this R^2 value for evaluating the quality of our energy model equation.

4. EXPERIMENTAL RESULTS

To demonstrate the validity of our approach for deriving an energy consumption model at the instruction level, we performed a set of experiments. In Section 4.1, we apply our proposed technique to derive the model equation for our target processor ARM7TDMI [1], using the cycle-accurate high-precision measurement hardware described in [4]. Then,

[1] We also introduce a set of *indicator* or *dummy* variables [5] that can only take a boolean value, to express the base cost for each instruction in the form of a linear combination.

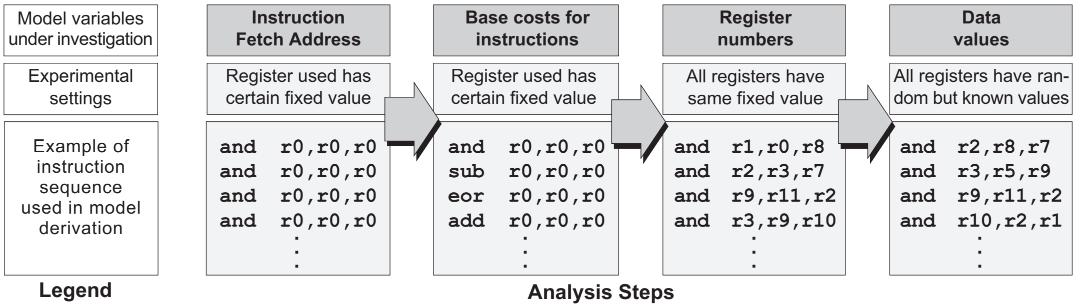

Figure 3: Stepwise derivation of characterizing parameters.

in Section 4.2, we estimate the energy consumption of a randomly-generated program using the model equation derived from our analysis.

4.1 Energy Model for ARM7TDMI Processor Core

We applied our proposed technique for energy model construction to the ARM7TDMI processor core. The target processor has a three-stage pipeline similar to the one shown in Figure 2. We derived an energy consumption model for the ARM data-processing instructions that have one of the following two instruction formats [7].

```
op    Rd, Rn, Rm
op    Rd, Rn, #imm
```

Here, `op` specifies the opcode for the instruction, and `Rd`, `Rn`, and `Rm` specifies the destination register, the first source operand register, and the second source operand register, respectively. Instead of the second source operand register, an immediate value can be designated as the second operand with the `#imm` field specified.

We defined a set of model variables at the instruction level, in terms of (1) instruction fetch address, (2) register numbers, (3) immediate operand, and (4) data values. We measured the energy consumption by executing a set of test programs on the target processor, to derive the values of various characterizing parameters in the model equation.

As previously mentioned in Section 3.2, we decompose the problem into several subproblems and derive the different characterizing parameters in a number of steps, for the sake of model accuracy and analysis simplicity. Figure 3 illustrates this problem decomposition using sample instruction sequences similar to the ones used in the derivation of characterizing parameters. In the first set of test programs, we executed the same instructions at different program locations with the same operand specifiers and the same data values, to derive the parameters for the Hamming distance and the weight of the instruction fetch address (denoted by FA). Using the regression analysis, we derived $H_{\mathrm{IF}}^{\mathrm{FA}/*} = 2.7164$ (pJ) and $W_{\mathrm{IF}}^{\mathrm{FA}/*} = -3.4378$ (pJ), which can be merged for all the instructions under consideration. In this case, the coefficient of determination was $R^2 = 0.9814$, which means that the model variable FA was capable of explaining more than 98 % of the energy variation due to the changes in the instruction fetch address. The results indicate that the energy consumption is proportional to the number of bit switches in the fetch address, and negatively proportional to the number of 1's in the fetch address. With this information available, we can apply program optimization for low energy such as code replacement, where frequently executed instructions are placed in program addresses with small Hamming distances and a large number of 1's.

Table 1: Base Costs for Different Instructions

Instruction	Pipeline Stage IF	ID	EX
`add`	300.7668	317.0254	327.9964
`sub`	298.1403	200.3411	189.3604
`and`	300.1020	272.6019	330.8285
`eor`	298.5258	419.6961	332.0663
$H_s^{\mathrm{op}/*}$	3.3085	11.7114	0.3015

(units: pJ, $R^2 = 0.9699$)

To derive the base costs for various instructions at each of the three pipeline stages, we executed another set of test programs. The test program consists of an instruction mix of different instructions, with the same operand specifiers and the same data values for all the instructions. To eliminate the effect of changes in the instruction fetch address, we calculated the energy variation due to the fetch address using the parameter values derived in the previous analysis step, and subtracted it from the measured energy value. Table 1 presents the results from our regression analysis for four sample instructions. [2] We also analyzed the impact of the opcode on the energy consumption, and the results show that the energy is proportional to the Hamming distance of the opcode, as shown in the last row of the table. The results indicate that the energy consumed at the IF stage is not substantially different from one instruction to another, as shown in the second column of the table. However, in stages ID and EX, the energy consumption of different instructions are substantially diverse. For example, the `sub` instruction consumes much less energy in the ID and the EX stages than the other instructions, while the `eor` instruction consumes much more energy in the ID stage. This large difference of the base costs for different instructions and the tendency of increase in the energy consumption with the

[2] Due to the space limitation, we only present the results for four instructions, although we derived the parameters for all the data-processing instructions in the ARM instruction set [7].

Table 2: Parameters for Register Numbers

Variable	Type	Pipeline Stage IF	ID	EX
Rd	H	0.8632	7.6405	0.8358
	W	0.6373	2.7714	0.5032
Rn	H	1.3448	3.9167	1.0192
	W	1.5510	3.4022	1.1363
Rm	H	0.8676	6.5098	1.4356
	W	2.2035	3.4690	−0.4734

(units: pJ, $R^2 = 0.9206$)

Table 3: Parameters for Data Values

Instruction	Variable	Type	Pipeline Stage ID	EX
add	src1	W	3.9338	6.6084
	src2	W	0.1757	7.3158
	dest	W	–	3.6336
sub	src1	W	8.8073	5.9504
	src2	W	9.5822	8.3584
	dest	W	–	4.8764
and	src1	W	8.6790	8.6559
	src2	W	0.4085	9.3909
	dest	W	–	−2.3031
eor	src1	W	2.7462	7.0900
	src2	W	−1.3005	7.5799
	dest	W	–	2.8201

(units: pJ, $R^2 = 0.9706$)

increase in the Hamming distance of the opcode indicates that the energy consumption of software can be reduced by applying instruction selection and/or instruction scheduling techniques tailored for low energy [18, 21]. Specifically, the energy consumption can be significantly reduced by selecting the instructions with low base costs when more than one choices are possible for the same execution semantics. Moreover, careful scheduling of instructions can save a substantial portion of energy consumed in the processor core, by reducing the number of bit switches in the opcodes of adjacent instructions. For example, reducing one opcode bit flip on average will save approximately 15.32 pJ of energy per instruction (the sum of savings in the three stages), which amounts to about 1.2 % of the total energy consumed in the processor core. Although this is a small fraction of the energy consumed by the processor core, much more energy savings are expected when we consider different parts of the system as well, e.g., instruction cache, main memory, and the interconnect bus.

To derive the characterizing parameters for register numbers, we executed still another set of test programs where the same instructions are executed with changes in the register numbers, i.e., `Rd`, `Rn`, and `Rm`. Again, to eliminate the effects of instruction fetch addresses, we calculated the variation due to the fetch address and subtracted it from the measurement data. Table 2 summarizes the results from our regression analysis. We successfully derived the parameter values for the Hamming distance and the weight of each register number in each of the pipeline stages, which can be merged for all the instructions under consideration. The results indicate that most of the energy variation due to the register numbers are in the ID stage, as can be induced from the large coefficient values in the fourth column of the table. Especially, the most significant factors are the Hamming distances of `Rd` and `Rm`, which have the largest coefficient values. This information can be used in program optimization techniques such as register assignment and/or register relabeling for low energy [3, 11]. Specifically, if we reduce one bit flip in each of the three register numbers on average, the resulting energy savings will be approximately 24.43 pJ per instruction, which corresponds to about 1.9 % of the total energy. Again, the expected energy savings by the techniques for reducing bit switches in the register numbers become much larger when we take into account the memory subsystem, since the register numbers affect the energy consumption of various levels of memory hierarchy and the interconnect bus as well.

Similarly, we executed a set of test programs with random data values to derive the coefficients for data values. Table 3 shows the results from our regression analysis for four sample instructions, where `src1` and `src2` denote the first and the second source operands, respectively, while `dest` denotes the destination operand. The results show that the variables defined in terms of data values have significance only in the weights, not in the Hamming distances. This phenomenon is presumably due to the *precharge-and-evaluation* scheme [4] used in the dynamic CMOS implementation of the target processor. Note that the weight of the destination operand in the ID stage is omitted in the table since the computation result will not be generated until the EX stage. Also note that, unlike the results from the previous analysis steps, different instructions have different parameter values for the model variables defined in terms of data values. This is due to the fact that different instructions behave differently with respect to data values, according to the operation specified by the instructions. We conjecture that this difference in energy behavior is due to the different ways of utilizing the functional units by different instructions.

Likewise, we performed regression analysis on the impacts of the immediate operand, which turned out to have only significance in the Hamming distance in the IF and ID stages. The coefficient values derived are $H_{\mathrm{IF}}^{\mathrm{imm}/*} = 4.0638$ (pJ), and $H_{\mathrm{ID}}^{\mathrm{imm}/*} = 8.9803$ (pJ), respectively, with the R^2 value of 0.9815.

4.2 Program Energy Estimation

To show the usefulness of our energy consumption model in estimating the energy consumed by an instruction sequence, we implemented an energy consumption analyzer. The analyzer takes an assembly source program (or a binary program) as its input, and estimates the energy consumption of the given instruction sequence in each clock cycle. The analyzer first extracts the values of the predictor variables by examining the instruction sequence. [3] Then it calculates the energy consumption at each clock cycle using the model

[3] In deriving the values of the predictor variables defined in terms of data values, we assumed that the initial values in the registers are known a priori.

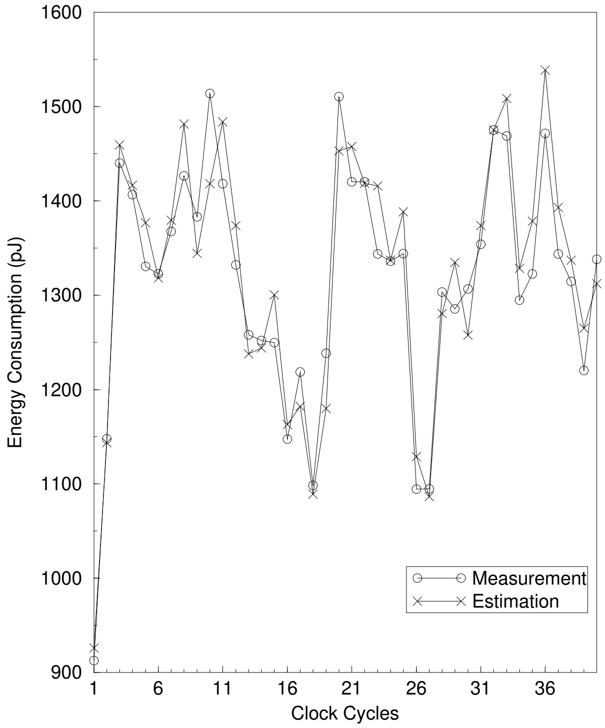

Figure 4: Comparison between estimated and measured energy.

equation and the values of the characterizing parameters previously determined by the regression analysis.

For the experiments, a sample program was generated that contains a random mixture of ARM data-processing instructions, with the operand specifiers, i.e., the register numbers and the immediates, are chosen at random. Also, the data values are randomly generated and prepared by the sample program prior to the execution of the instruction sequence under investigation. Figure 4 compares the estimated energy with the actual energy consumption measured on the hardware, for the same sample instruction sequence that is executed for 40 clock cycles.

The results show that the proposed approach provides an accurate energy estimation, which is verified by the fact that the graph for the estimated energy closely resembles that of the measured energy. The coefficient of determination (R^2) value in this case is 0.9893, which means that more than 98 % of the variation in the measured energy is captured by our estimation equation. The total of the measured energy consumption for the 40 clock cycles is 5.2352 nJ, while that of the estimated energy is 5.2993 nJ. The error of this total energy is less than 1 %. We also calculated the error ratio in each cycle given by

$$r_e = \frac{|\text{estimation} - \text{measurement}|}{\text{measurement}}\ , \tag{7}$$

whose average value is 0.0251, which means that the error of the energy estimation by our model equation was on average 2.5 % of the total energy. Besides, the maximum of the error ratio is 0.0633, which means the error was at most 6.33 %. The error comes from a number of sources, including the following.

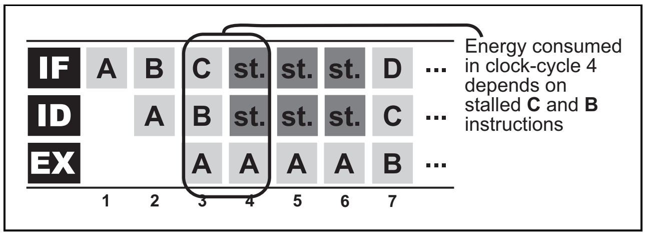

Figure 5: Example of pipeline stalls.

- The energy consumption may not be perfectly linear to the Hamming distances and the weights of the model variables.
- The model might have missed one or more of variables that have significant impacts on energy consumption.
- The quantization error inherent in the measurement hardware used in the model construction may cause inaccuracies in deriving the characterizing parameters.

5. CONCLUSIONS AND FUTURE WORK

We have proposed a technique for deriving an accurate energy consumption model at the instruction level. The proposed approach combines an empirical method and a statistical analysis technique called regression analysis. The result of the proposed approach for deriving the energy model is given in the form of a model equation, which can be used to estimate the energy consumption of a given instruction sequence using the properties of the instructions. The energy consumption model described in this paper enables a simple but yet accurate method for estimating the energy consumption at the instruction level. The availability of this instruction-level energy model will provide a basis for various energy reduction techniques for embedded software, by identifying the factors that contribute to energy consumption, and further indicating the significance of each factor. Another advantage of the proposed approach is that the model derivation can be retargeted to a wide variety of processors, since it does not require information about the processor implementation. In other words, the proposed technique can be used to derive energy model equation for an arbitrary processor, provided that a mechanism is available for obtaining cycle-level energy consumption data. For this purpose, an energy simulator can be used as well as a measurement hardware similar to the one used in this paper.

We applied our proposed method to derive an energy consumption model for the ARM7TDMI processor core, to verify the validity of the approach. The case study shows that an accurate energy model can be derived from instruction-level characterization of software, using the technique proposed in this paper. The results from our experiments indicate that our energy model can accurately estimate the energy consumption of a given instruction sequence. The statistical results show that more than 98 % of the energy variation can be explained by the derived model equation. The maximum error ratio of the prediction was 6.3 %, while the average was approximately 2.5 %.

We are currently investigating a technique for analyzing the effect of pipeline stalls, as shown in Figure 5. In the figure, instructions `B` and `C` are stalled in the pipeline stages

ID and IF, respectively, because instruction `A` executes for more than one clock cycle in the EX stage. To analyze the energy consumption in the case where one or more instructions are stalled in the pipeline, we augment the model by defining the stall state of each pipeline stage as a special case where no instruction is occupying the stage. With this extension, we expect to be able to model the energy consumption of multi-cycle instructions such as a multiply instruction, whose execution cycle in the EX stage is dependent on the data values that it operates on. In addition, it will allow us to model the stalls due to the pipeline flushes by branch instructions and pipeline hazards due to various reasons such as data dependency and resource conflicts.

We are also investigating a technique for modeling load/store instructions. Since the energy consumed by the execution of these memory reference instructions is dependent on whether the access hits or misses in the cache, we need to integrate a technique for analyzing the cache behavior. Moreover, since the energy consumption is also dependent on the characteristics of memory devices, we are checking the feasibility of incorporating an empirical or analytical energy consumption model for memory devices.

6. ACKNOWLEDGMENTS

The authors would like to thank Jakob Engblom and Hans Hansson for their fruitful comments on drafts of this article as well as Giancarlo Iannizzotto for inspiring discussions on the analysis methodology. The comments from the anonymous reviewers further improved the quality.

7. ADDITIONAL AUTHORS

Additional authors: Naehyuck Chang (School of Computer Science and Engieering, Seoul National University, email: `naehyuck@snu.ac.kr`).

8. REFERENCES

[1] Advanced RISC Machines Ltd. *ARM7TDMI Data Sheet*, August 1995.

[2] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto. An instruction-level functionality-based energy estimation model for 32-bits microprocessors. In *Proceedings of the Annual ACM IEEE Design Automation Conference*, pages 346–351, June 2000.

[3] J.-M. Chang and M. Pedram. Register allocation and binding for low power. In *Proceedings of the Annual ACM IEEE Design Automation Conference*, pages 29–35, June 1995.

[4] N. Chang, K.-H. Kim, and H. G. Lee. Cycle-accurate energy consumption measurement and analysis: Case study of ARM7TDMI. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 185–190, July 2000.

[5] S. Chatterjee, A. S. Hadi, and B. Price. *Regression Analysis by Example, Third Edition.* John Wiley & Sons, Inc, 2000. ISBN 0-471-31946-5.

[6] R. Y. Chen, M. J. Irwin, and R. S. Bajwa. An architectural level power estimator. In *Proceedings of the Power-Driven Microarchitecture Workshop*, 1998.

[7] S. Furber. *ARM System Architecture.* Addison-Wesley, 1996. ISBN 0-201-40352-8.

[8] C. Gebotys, R. Gebotys, and S. Wiratunga. Power minimization derived from architectural-usage of VLIW processors. In *Proceedings of the Annual ACM IEEE Design Automation Conference*, pages 308–311, June 2000.

[9] C. H. Gebotys and R. J. Gebotys. An empirical comparison of algorithmic, instruction, and architectural power prediction models for high performance embedded DSP processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, 1998.

[10] M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and W. Ye. Influence of compiler optimizations on system power. In *Proceedings of the Annual ACM IEEE Design Automation Conference*, pages 304–307, June 2000.

[11] M. Kandemir, N. Vijaykrishnan, M. J. Irwin, W. Ye, and I. Demirkiran. Register relabeling: A post-compilation technique for energy reduction. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power*, October 2000.

[12] B. Klass, D. E. Thomas, H. Schmit, and D. F. Nagle. Modeling inter-instruction energy effects in a digital signal processor. In *Proceedings of the Power-Driven Microarchitecture Workshop*, June 1998.

[13] M. T.-C. Lee, V. Tiwari, S. Malik, and M. Fujita. Power analysis and minimization techniques for embedded DSP software. *IEEE Transactions on VLSI Systems*, pages 1–14, March 1997.

[14] H. Mehta, R. M. Owens, and M. J. Irwin. Instruction level power profiling. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1996.

[15] H. Mehta, R. M. Owens, M. J. Irwin, R. Chen, and D. Ghosh. Techniques for low energy software. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 72–75, August 1997.

[16] 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*, October 1998.

[17] M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria. Instruction-level power estimation for embedded VLIW cores. In *Proceedings of the International Conference on Hardware Software Codesign*, pages 34–38, May 2000.

[18] C.-L. Su, C.-Y. Tsui, and A. M. Despain. Low power architecture design and compilation techniques for high-performance processors. In *Proceedings of the IEEE COMPCON*, pages 489–498, 1994.

[19] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. *IEEE Transactions on VLSI Systems*, 2(4):437–445, December 1994.

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

[21] M. C. Toburen and T. M. Conte. Instruction scheduling for low power dissipation in high performance microprocessors. In *Proceedings of the Power-Driven Microarchitecture Workshop*, June 1998.

[22] W. Ye, N. Vijaykrishnan, 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*, pages 340–345, June 2000.